

TS1000-ZX81 OWNERS WIN \$20,000

ENTER THE ULTIMATE ADVENTURE



SOLVE 12 CLUES LIKE THIS!

*Where it all began. Where the torch was first lit.
Where muscles and sinews strain. Where our heros
win acclaim. Where the symbols hold the key.*

KRAKIT™ consists of 12 clues on a ready-to-run ZX81 or TS1000 cassette tape (16k RAM). The answer to each clue is the name of a country, a city or town, and a number. If you are the first qualified entrant to solve all 12 clues and declared the winner, you receive two tickets to the city of the secret KRAKIT™ vault location. When you arrive at that location, a check for a minimum amount of \$20,000.00 (U.S.) will be presented to you. The amount of the prize money is augmented weekly.

TS1000-ZX81

SYNTAX QUARTERLY/SPRING 1983



Mail to:
INTERNATIONAL PUBLISHING & SOFTWARE INC.
P.O. BOX 1654, BUFFALO, N.Y. 14216

Please
at
Total

Char
Num
E

USE THAT POWER

A PUBLICATION OF
THE HARVARD GROUP

SQ

SYNTAX QUARTERLY

Serving The Timex-Sinclair Family of Personal Computers

Spring 1983

Vol. 2, No. 1

SQ Staff

Publisher

Kirtland H. Olson, P.E.

Editor

Ann L. Zevnik

Assistant Editor

Pamela Petrakos-Wilson

Technical Consultant

Eric K. Olson

Editorial Assistants

Pamela K. Vinal

Karen Brody

Production

CSA Press

Circulation

Susan C. Shippen

Constance Saarinen

Timothy Parker

Financial Manager

Mary Russo

Designer

RDS Designs

Cover Photography

James Rue Design

Contents

Syncwars—An Invasion	14
The invasion of Computerburg. <i>by Fred Nachbaur</i>	
Geography Review—U.S. States Quiz	19
A review of U.S. geography. <i>by Steven Walley</i>	
POKEing Directly to the Display File	22
Use less memory, speed up graphics. <i>by Dan Platt</i>	
Simultaneous Linear Equations and Matrix Inversion	24
Solve equations and invert matrices with this program—a delight for amateur mathematicians. <i>by R.A. Woodall</i>	
ZX/TS Home Budget	26
Home finances on your personal computer. <i>by Michael Roberts</i>	
Getting Your Head Straight	30
Align the heads on your recorder. <i>by John Andrews</i>	
Make-It-Yourself Quiz Program	34
Choose your format, tailor questions and answers. <i>by Dale F. Lipinski</i>	
Build Your Own EPROM Programmer and Centronics Printer Interface—Part II	38
Build 2 EPROM read boards and a 2-way Centronics compatible parallel printer port with step-by-step instructions. <i>by John Olinger</i>	
Operation Codes of the 8080, 8085 and Z80 Processors	46
A complete reference to the op-code mnemonics of 8080, 8085 and Z80 microprocessors. <i>by D Martin Harrell</i>	

Departments

s' BASIC	4
Reviews	8
Reviews	54
ws	57
	61
	64

This Could be Your Chance for Fame and Fortune (or at least fame)

Would you like to see your name in print? **SQ** and **SYNTAX** newsletter are looking for authors. If you have a program or article you'd like to share with other ZX/TS owners, send it along.

Our requirements are simple: all we ask is that your program run with no bugs and your article present accurate information.

However, here are a few more details that enhance your chances of being published:

PROGRAMS: Any program for a fun or useful purpose is a candidate for **SYNTAX** or **SQ**. We publish game, business, educational, utility and science/math programs. Please send programs on cassette, particularly if they're long. We print programs directly from a Sinclair printer to minimize errors. If you send your program on cassette, it will appear exactly as you entered it. A hard, or paper, copy helps us figure out any problems that may arise. Also include as much documentation as you can, including a list of variables used and their functions. Write complete instructions for using your program so the most inexperienced user could not mess it up. Sample output also helps, especially if your program generates charts, graphs, or numerical tables. Make your program as user-friendly as possible within the limitations of available memory. Indicate how much memory your program requires. And include the Syntactic Sum of the program as listed. For a copy of the Syntactic Sum program, contact **SYNTAX** (address following).

ARTICLES: Articles can cover virtually any topic related to ZX/TS computers and their applications. Our readers run the gamut from expert to novice, so we are looking for articles written at all levels of complexity. Your article can present a

programming tip, explain the function of the machine or a command, or describe a hardware project. If you present ideas on programming, it helps to include short sample routines demonstrating your ideas. If your article discusses hardware, please include sketches, schematics, or even photos to help readers follow along.

For both programs and articles, type your submission or print clearly, preferably double-spaced. Put your name, address and day-time phone number on the first page. If you want us to return your material to you, include a self-addressed stamped envelope.

You can submit to either **SYNTAX** newsletter or **SQ** magazine. Generally, programs and articles in **SQ** tend to be longer and more involved than those in **SYNTAX** because of space considerations. Which publication your submission ultimately appears in is up to the editors.

What's in it for you? Aside from the fame and glory of appearing in **SYNTAX** or **SQ**, you'll receive a check from us dependent on the length of the published material. We pay 7 cents for each 6 characters, including spaces and punctuation. This payment buys us the nonexclusive right to use your material. We can use it and so can you, so you can sell it again to anyone you like. Programs receive a token \$2 payment each. Not enough to retire on, but often enough to pay for your subscription.

So dig through your notes and files and send off anything you think may help other users use their computers better. Address submissions to:

Editor
SYNTAX/SQ
Rd 2, Box 457
Harvard, MA 01451
USA

\$5.95

8 1/2" x 11" credit-card-type-plastic
MICRO CHARTS are fact-packed
microprocessor summary cards ...
that save you time and effort with
instant access to the data you need.

Clear and concise tables for: full instruction set, disassembly, ASCII, base conversion, effect on flags, inequalities versus jumps, interrupt structure, pinout, cycle times, diagrams, notes, and much more.

Thousands of delighted users have made the switch from black & white, incomplete, folded-up, paper summaries with scrambled data ... to unscrambled, comprehensive, ready-to-use **MICRO CHARTS**.



Two sided colorful **MICRO CHARTS** are ideal for professionals, hobbyists, and students alike. Countless reorders and repeated praise confirm the benefit of these inexpensive, improved programming, debugging and learning aids.

Buy one. Or the whole set. You'll never switch back. We guarantee it. And you'll soon see that lifetime **MICRO CHARTS** are worth much more ... than the cost of a couple of hamburgers! Order now.

Syntax ZX80 Inc. (617) 456-3661, Dept. 1, RD 2, Box 457, Harvard, MA 01451

YES, please rush the **MICRO CHARTS** indicated below at just \$5.95 each. Unless I am 100% pleased in 14 days, a prompt full refund will be made.

Quan.	Chart Title	Total Quan.
	Z80 CPU	
	6502 (65XX)	Sub Total
	8080A & 8085A	\$
	8048 & RELATIVES	Pstg & Hndg
	ALGORITHMS	\$1.00

☐ Save \$5.80! Select any 5 for only \$24.95 postpaid (10 at \$49.90 etc.). Same 100% money back guarantee.

Total Enclosed
\$

YOUR NAME

COMPANY

ADDRESS

CITY

STATE

ZIP

SQ-ups—Winter 82

Getting Information into Your Computer by Jim Conrad
On p. 6, column 2, line 10 of the first example program should read:

```
10 LET B=17
```

In the table on p. 8, under "Example" the third entry should read Reserved Words (TO, RUN, TAN).

On p. 9, column 2, line 10 should read:

```
10 PRINT "first message"
```

Improved Super Monzzer Game by Robert Calhoon
Line 240 should read:

```
240 PLOT 23,21
```

The last word in line 1700 is correctly spelled MONZXER. We apologize for incorrectly spelling Mr. Calhoon's name.

Syntactic Sum with Variables by Donald J. Beck

In the assembly listing on pp. 18-19, all appearances of the number 16346 should read 16396 and all appearances of the number 16347 should read 16397. The instances occur at addresses:

32720 (mnemonic and comment)

32721 (comment)

32722 (comment)

32723 (comment)

The code at address 32743 should be 229.

Machine Code Programs—Where and How to Load Them by William Wentz

On page 21, first paragraph under Variables Storage Area, the last sentence should read: For ease in placing the MC into the string and locating the USR address, put the LET statement containing the MC first in the program, so it will be the first variable in storage.

ZX/TS Directory Program by David R. Rowland

The variables in line 1, 2, and 3 should all be dimensioned to (90,31), not (1,31) to allow all 90 entries. See reprinted listing following.

Keyboard Conversions by David M. Straub

In Figure 1, p. 27, every key should have two black dots indicating pins. Those in the upper right corner are missing one dot.

Using Extra Keys on Big Keyboard—A Simpler Approach by David Ornstein

Please note that the anode of D3 indicated on the schematic is for a ZX80. On a ZX81 or TS1000, that diode is D6.

Build Your Own EPROM Programmer and Centronics Printer Interface by John Oliger

Except when referring to ROM installation inside the ZX/TS, Dn always means Dn' (which appears at the edge connector) in parts I and II of this article. Dn means data line n.

Program Clarification

Parts of programs in SQ Winter 82 did not print as clearly as we would like. Here are portions of those listings reprinted for easier reading.

A Program to Index Articles

```
10 DIM A$(500)
20 DIM B$(400,24)
30 DIM C$(400,5)
80 PRINT AT 8,6;"SYNTAX INDEX"
120 PRINT AT 9,10;"1. APPEND"
130 PRINT AT 10,10;"2. LIST"
140 PAUSE 50000
150 POKE 16437,255
160 IF INKEY$="1" OR INKEY$="A"
THEN GOTO 1000
:
:
220 INPUT K
230 IF K<1 OR K>6 THEN GOTO 220
235 CLS
240 GOTO K*50+250
250 SCROLL
260 PRINT " ";C$(I);"-";B$(I)
265 SCROLL
270 PRINT "VOL/ISSUE/PAGE ";A$(I);B$(I);C$(I)
272 SCROLL
275 PAUSE 90
280 IF INKEY$="P" THEN PAUSE 50000
285 POKE 16437,255
290 RETURN
300 FOR I=1 TO 400
305 LET A=INT ((CODE A$(I)-1)/12)
310 LET B=CODE A$(I)-A*12
315 LET C=CODE A$(I+400)
320 GOSUB 250
325 NEXT I
```

Improved Super Monzzer

The words in reverse video are:

2030 HURRAY HURRAY

```
1420 PRINT "THERE IS A SPIDER IN
YOUR ROOM AND HE IS HUNGRY"
1423 PRINT AT 9,10;" / / "
1424 PRINT AT 10,9;" "
1425 PRINT AT 11,9;" "
1427 PRINT AT 12,17;"IL IL IL"
1428 PRINT AT 14,7;"YUM YUM EATE
N UP"
1429 PAUSE 100
1430 FOR I=1 TO 10
1440 IF YR=P(I) THEN GOTO 1470
1450 NEXT I
1460 GOTO 1540
1470 IF YL=5 THEN GOTO 1520
```

Continued Next Page

```

1480 CLS
1490 PRINT "TSK, TSK, YOU FELL I
N A PIT"
1500 LET YR=YR+20
1505 IF YL>=2 THEN GOTO 1540

...

3030 LET K$=INKEY$
3040 IF K$="Y" THEN GOTO 3070
3050 IF K$="N" THEN GOTO 3130
3060 GOTO 3020
3070 PRINT AT 10,6;"SAME TUNNEL
SET?"
3080 PAUSE 900
3090 LET K$=INKEY$
3100 IF K$="Y" THEN GOTO 525
3110 IF K$="N" THEN GOTO 399
3120 GOTO 3080
3130 CLS
3140 PRINT AT 10,20;"HAR HAR"
3150 PRINT AT 12,18;"YOU LOSE"
3160 LET END=1
3165 FAST
3170 GOTO 70
3500 PRINT AT 10,9;"TT TT TT"
3510 PRINT AT 11,8;" / "
3520 PRINT AT 12,5;">>=== "
3530 PRINT AT 13,11;" / "
3540 PRINT AT 14,12;" "
3542 PRINT AT 9,20;"? ?"
3544 PRINT AT 7,24;"? "
3546 PRINT AT 5,27;"? "
3550 PRINT AT 16,22;"YA GOT ME"
3560 RETURN

```

MC Programs—Where and How to Load Them

```

10 REM 12ELANDLN TAN
20 LET H$="....."
40 LET N=LEN H$/3
50 LET D=PEEK 16396+256*PEEK 1
6397-2
60 FOR I=1 TO N
70 POKE 16515,INT ((D+I)/256)
80 POKE 16514,D+I-256*PEEK 165
15
90 RAND USR 16516
100 POKE D+I,16*CODE H$(I*3-2)+
CODE H$(I*3-1)-476
110 NEXT I
120 POKE D-1,INT ((N+2)/256)
130 POKE D-2,N+2-256*PEEK (D-1)
200 REM
SYNTACTIC SUM: 20307, 8K ROM

```

```

10 REM 12ELANDLN TAN
20 LET D$="....."
40 LET N=LEN D$/3
50 LET D=PEEK 16396+256*PEEK 1
6397-2
60 FOR I=1 TO N
70 POKE 16515,INT ((D+I)/256)
80 POKE 16514,D+I-256*PEEK 165
15
90 RAND USR 16516
100 POKE D+I,VAL D$(I*4-3 TO I*
4-1)
110 NEXT I
120 POKE D-1,INT ((N+2)/256)
130 POKE D-2,N+2-256*PEEK (D-1)
200 REM
SYNTACTIC SUM: 19287, 8K ROM

```

```

10 REM LN 7?E,RND GOSUB ???T
GOSUB ??RND GOSUB ? FOR ; GOSUB
GOSUB ??RND,RND GOSUB PI6,RN
DE,RND GOSUB PI6,RND GOSUB ?T GC
SUB PI RAND TAN
20 LET H$="....."
40 LET L=LEN H$/3
50 LET M=L
60 IF L<20 THEN LET M=20
70 POKE 16507,M
80 PRINT USR 16514;" BYTES SAV
ED"
90 LET N=PEEK 16388+256*PEEK 1
6389
100 PRINT "RECORD USR ADDRESS "
;N
110 FOR X=3 TO L*3 STEP 3
120 POKE N,16*CODE H$(X-2)+CODE
H$(X-1)-476
130 LET N=N+1
140 NEXT X
150 PRINT " MACHINE CODE LOADED
"
SYNTACTIC SUM: 23944, 8K ROM

```

ZX/TS Directory Program

The words in reverse video are:

62 EVER AGAIN
196 MEMORY FULL
197 CONT
455 NONE FOUND
457 CONT
490 CONT
915 the Y in DIRECTORY

```

1 DIM N$(90,31)
2 DIM S$(90,31)
3 DIM T$(90,31)
6 REM START PROGRAM WITH "GO
TO 16" AFTER INITIAL RUN
9 LET D=0
10 PRINT AT 0,12;"HELLO";AT 2,
1;"ENTER CODE FOR WHAT IS WANTED"
11 PRINT AT 4,0;"1. ENTER MOR
E NAMES"
12 PRINT "2. CHANGE SELECTED
NAME OR OTHER DATA"
13 PRINT "3. CHECK ALL ENTRIE
S"
14 PRINT "4. LIST ALL NAMES B
EGINNING"," WITH A LETTER"
15 PRINT "5. LIST ALL NAMES I
N STATE"
16 PRINT "6. LIST ALL NAMES N
OT IN STATE"
17 PRINT "7. LIST ALL NAMES B
Y ZIP CODE"
18 PRINT "8. LIST ALL CODED N
AMES"
19 PRINT "9. SAVE REVISED DIR
ECTORY"

```

```

160 INPUT J$
165 CLS
170 IF J$="Y" THEN GOSUB 750
172 PRINT AT 0,0;L,,N$(L),S$(L)
,T$(L),,,
175 PRINT "ALL CORRECT? Y OR N
"
178 INPUT J$
180 IF J$<>"N" THEN GOTO 184
181 LET A=1
182 GOSUB 330
183 LET A=0

```

Continued Next Page


```

184 CLS
185 PRINT AT 6,0;"ANY MORE NAME
S? Y OR N."
187 INPUT U$
190 CLS
192 IF U$<>"Y" THEN RETURN
195 NEXT L
:
:
340 INPUT Q
350 IF Q=1 THEN INPUT N$(L)
355 IF Q=2 THEN INPUT S$(L, TO
19)
360 IF Q=3 THEN INPUT T$(L, TO
19)
365 IF Q=4 THEN INPUT T$(L,23 T
0 25)
370 IF Q=5 THEN INPUT T$(L,27 T
0 31)
375 IF Q=6 THEN INPUT S$(L,20 T
0 )
380 IF Q=7 THEN GOSUB 750
385 IF Q=8 THEN GOTO 35
388 IF Q=9 THEN GOSUB 660
389 CLS
395 GOTO 305
400 PRINT AT 4,0;"ENTER LETTER (
S) OR NAME TO BE","LOCATED"
405 LET Q=0
410 INPUT U$
420 CLS
430 FOR L=1 TO D
440 IF U$=N$(L, TO LEN U$) THEN
GOSUB 465
450 NEXT L
455 IF Q=0 THEN PRINT AT 7,10;"
HERE HERE
457 PRINT AT 20,0;"PRESS ""HERE
"" TO REGAIN MENU"
460 STOP
462 RETURN
465 PRINT L,,N$(L),S$(L),T$(L),
'
470 LET Q=Q+1
480 IF INT (Q/4)=Q/4 THEN GOTO
490
485 RETURN
490 PRINT AT 20,0;"PRESS ""HERE
"" TO RESUME LISTING"
492 STOP
495 RETURN
497 GOTO 450
500 PRINT AT 5,0;"ENTER TWO LET
TER STATE CODE"
505 LET Q=0
510 INPUT U$
515 CLS
520 FOR L=1 TO D
525 IF U$=T$(L,23 TO 24) THEN G
OSUB 465
530 NEXT L
540 GOSUB 455
550 RETURN
600 PRINT AT 5,0;"ENTER TWO LET
TER STATE CODE"
605 LET Q=0
610 INPUT U$
615 CLS
620 FOR L=1 TO D
625 IF U$<>T$(L,23 TO 24) THEN
GOSUB 465
630 NEXT L
640 GOSUB 455
650 RETURN
660 LET N$(L)=N$(D)
665 LET S$(L)=S$(D)
670 LET T$(L)=T$(D)
675 LET N$(D)=" "
680 LET S$(D)=" "
685 LET T$(D)=" "
690 LET D=D-1
695 RETURN
700 PRINT AT 5,0;"ENTER 5 NUMBE
R ZIP"

```

SINWARE provides high-quality machine-code programs for the TS1000 or ZX81.

HOT Z

HOT Z is the machine-programming editor, debugger and disassembler that takes the mystery out of assembly language. Over 40 cursor-driven commands give you an interactive system for entering, revising and relocating code. Full-screen listings with your labels let you understand other programs and capture the power of ROM routines for your own programs.

SQ said of HOT Z: *"Easily the best machine language debugging package I have ever seen for the ZX81. . . If you program in machine language and need the best tool for the job, buy HOT Z."*

HOT Z is just \$19.95 + \$2.00 pp on cassette in different versions for 16K or 32K+. Please specify. **NOW ON EPROM** (four 2716's) mapped to the 8-16K block for the Hunter or similar board, HOT Z-E is \$40.

Z EXTRA

Z EXTRA is a display creator/controller that makes you a master of ZX graphics and displays. No programming is required to create, save, print or display multiple screens of text and graphics. Z EXTRA features a full-screen editor, blinking cursors, repeating keys, four write directions, eight plot directions, 4x4 and 8x8 character sizes, and much more.

Z EXTRA's displays provide horizontal or vertical scrolls of multiple screens against a background screen, or timed page flips for simple animation. Screens can be transferred to BASIC strings to save hours of fussy programming.

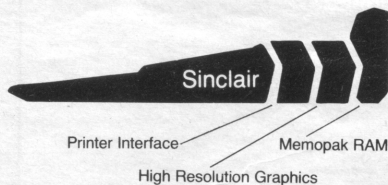
Z EXTRA requires a ZX81 or TS1000 with at least 16K of RAM and is especially useful with 64K. Just \$19.95 + \$2.00 pp on cassette.

SINWARE
BOX 323, DIXON, NM 87527



BEHIND EVERY GOOD SINCLAIR IS A MEMOPAK

If you own a Timex-Sinclair 1000 or ZX81 computer, you should have a Memopak behind it. From increased memory to high resolution graphics, Memotech has a Memopak to boost your system's capabilities. Every Memopak peripheral comes in a black anodised aluminum case and is designed to fit together in "piggy back" fashion to enable you to continue to add on and still keep an integrated system look.



Order at no risk

All Memotech products carry our 10 day money back guarantee. If you're not completely satisfied, return it in ten days and we will give you a full refund. And every Memotech product comes with a six month warranty. Should anything be defective with your Memopak, return it to us and we will repair or replace it free of charge. Dealer inquiries welcome. To order any Memotech product call our toll-free number **800/662-0949** or use the order coupon.

MEMOTECH
CORPORATION

7550 West Yale Avenue
Denver, Colorado 80227
(303) 986-1516
TWX 910-320-2917



Memopak 64K RAM The 64K RAM extends the memory of your Sinclair by 56K to a full 64K. It is directly addressable, user transparent, is neither switched nor paged and accepts such BASIC commands as 10 DIM A (9000). The Memopak 64K turns your Sinclair into a powerful computer suitable for business, recreational and educational use. No additional power supply is required.

Memopak 32K RAM The 32K RAM Memopak offers your Sinclair a full 32K of directly addressable RAM. Like the 64K Memopak, it is neither switched nor paged and enables you to execute sophisticated programs and store large data bases. It is also fully compatible with Sinclair's or Memotech's 16K RAM to give you a full 48K of RAM.

Memopak 16K RAM The Memopak 16K RAM provides an economical way to increase the capabilities of your Sinclair. And at the same time, it enables you to continue to add on other features with its "piggy back" connectors. It is compatible with the Sinclair 16K or a second Memopak 16K or Memopak 32K to give 32K or 48K of RAM respectively.

Memopak High Resolution Graphics The Memopak HRG contains a 2K EPROM monitor and is fully programmable for high resolution graphics. The HRG provides for up to 192 by 248 pixel resolution.

Memopak Printer Interface The Memopak Centronics Parallel or RS232 Interface packs enable your Sinclair to use a wide range of compatible printers (major manufacturers' printers available through Memotech at significant savings). The resident software in the units gives the ASCII set of characters. Both Memopak printer interfaces provide lower case character capabilities. The RS232 Interface is also compatible with modems.

New products coming soon Memotech will soon be introducing four new Sinclair compatible products: a high quality, direct connection keyboard, a digitizing tablet, a 16K EPROM and a disk drive. Watch for our future advertisements.

Mail to: Memotech Corporation, 7550 West Yale Ave., Denver, CO 80227			
Code: SQ2			
	*Price	Qty.	Total
64K RAM	\$179.95		
32K RAM	109.95		
16K RAM	59.95		
Centronics Parallel Printer Interface	104.95		
RS232 Printer Interface	139.95		
High Resolution Graphics	144.95		
Shipping and handling	4.95		\$4.95
* All prices quoted in U.S. dollars		Tax**	
** Colorado residents please add sales tax		Total	
☐ Check ☐ MasterCard ☐ Visa			
Account No.			Exp.
Name			
Address			
City	State	Zip	

Exploring String Functions

by James A. Conrad, Seattle, WA

Most novice programmers don't understand much about string functions. And many not-so-novice programmers don't use them to their full potential. Functions seemed relatively unimportant when they started learning BASIC—something to “get to later.” Well, “later” is now. String functions are easy to understand.

String functions are especially useful in writing input-checking and output-formatting routines—areas where many programs are weak. A few hours invested in studying string functions will pay dividends in user-friendly programs.

Many programmers tend to confuse functions and statements. A statement is an instruction to the computer, telling it to do something. It contains (or is) a verb, such as PRINT or GOTO. Most functions calculate or convert. They're like self-contained subroutines; SQR x, for example, computes the square root of x.

The major BASIC functions perform arithmetic or string operations. String functions analyze and manipulate strings. (For a refresher on strings, see SQ Winter 1982.—Ed.) They're indispensable in input-checking and output-formatting routines.

A Few Words about Functions

Functions have two parts: a title and an argument. The title describes, in BASIC, what the function does, for example, LENgth, VALue. The argument follows the title and serves as the input to the function. The function uses this input and returns a result. A function's argument is pronounced “of”—to read the function LEN A\$, for example, you say, “length of A string.”

You can do almost anything with a function that you can with a variable. You can print it. If it's numeric, you can add or subtract it. You can even put it in the argument of another function, for example, LEN A\$(2 TO 5). (This is called nesting. When you nest functions, be sure you put a right parenthesis for every left one.) About the only thing you can't do with most functions that you can do with variables is assign values to them with assignment statements (see SQ, Winter 1982).

The String Functions

First, a few definitions relating to string functions:

String: a set of alphanumeric and/or graphic characters.

String variable: name given to the memory cell(s) to which the string is assigned. Your ZX/TS gives you 26 string variables, A\$ through Z\$.

Substring: portion of a string consisting of consecutive characters.

Expression: a constant (such as 12 or “BOB”), a variable (X, Y\$), or a formula (7*Y, K\$+“DAVE”). If a formula is used in the argument of a function, you must put it in parentheses.

Here's a summary of string functions, what they do, and a program line that prints an example. The symbols I use in the arguments are:

\$:	string being analyzed
sub\$:	substring
len:	length of string or substring
pos:	position in string
x:	any numeric expression or number the computer can handle
chr:	character (in quotation marks) or character code
cod:	character code (0-255)

Any of these except *chr* can be a constant, variable or formula.

We'll set up line 10 to make a nine-character string A\$ to play with. Enter line 10 on your computer:

```
10 LET A$="123456789"
```

Slicing

Slicing is a process that gets substrings from strings. For the academically inclined, slicing is a function without a title—the entire slicing notation is the argument. The general form of the slicing notation is:

string expression (slicer)

The slicer must be enclosed in parentheses. Its general form is:

(position 1 TO position 2)

It also has several abbreviated forms, shown following:

Here is the full form for slicing. It returns a substring of string \$, beginning with character number *pos1* and ending with the character in *pos2*:

\$(pos1 TO pos2)

Enter line 20 and RUN our two-line program. (Note that you don't need to type in the spaces around TO. Your ZX/TS does that automatically when you use SHIFT 4 to enter TO).

```
20 PRINT "CHARACTERS 2, 3, AND  
4 OF A$ ARE: ";A$(2 TO 4)
```

You'll see 234 on your screen.

Now some shorthand notations:

\$(TO pos2) *pos1 omitted* returns the left portion of string \$ through *pos2*.

Add line 30:

SQ BEGINNERS' BASIC

```
30 PRINT "THE FIRST 4 CHARACTERS OF A$ ARE: ";A$(1 TO 4)
```

This prints 1234.

$\$(pos1 TO)$ *pos2 omitted* returns the right portion of string \$ from *pos1* through the final character. Add line 40 to our example program:

```
40 PRINT "THE RIGHT SIDE OF A$ FROM CHARACTER 5 IS: ";A$(5 TO )
```

Line 40 will print 56789.

$\$(pos n)$ or $\$(pos n TO pos n)$ gives you the character at *pos n* in string \$. Try line 50:

```
50 PRINT "THE 4TH CHARACTER OF A$ IS: ";A$(4)
```

$\$()$ or $\$(TO)$ gives you the entire string \$ as a substring. Line 60 demonstrates:

```
60 PRINT "A$ IS: ";A$()
```

It prints 123456789.

LEN \$ returns the number of characters in string \$. Try line 70, remembering that *LEN* is a function (press SHIFT ENTER, then the K key):

```
70 PRINT "NUMBER OF CHARACTERS IN A$ IS: ";LEN A$
```

You'll get 9.

VAL \$ determines the numeric value of string \$. Line 80 shows this (*VAL* is a function on the J key):

```
80 PRINT "THE VALUE OF A$ IS: ";VAL A$
```

We get 123456790. Note the computer rounded our nine-digit number to eight significant digits.

STR\$ X converts a numeric expression *X* into a string. Try lines 90 and 100 (*STR\$* is a function on the Y key):

```
90 LET A=333
100 PRINT "THE STRING OF A IS: ";STR$ A
```

This prints 333.

CODE \$ returns the character code number of the first character of string \$. You'll get zero if string \$ is a null, or empty, string or if the first character is a blank space (character code 0). Line 110 demonstrates (*CODE* is a function on the I key):

```
110 PRINT "A$ BEGINS WITH CHARACTER CODE NUMBER: ";CODE A$
```

Line 110 prints 29—the character code for 1 (check your manual). Try just entering PRINT CODE "1". You'll also get 29.

CHR\$ cod gives you the character, in a one-character string, corresponding to the character code *cod*. Remember, *cod* can be an expression (constant, variable or formula) whose value falls between 0 and 255. *CHR\$* is a function on the U key. Enter line 120:

```
120 PRINT "THE CHARACTER FOR CODE 29 IS: ";CHR$(29)
```

This prints 1.

Workhorse

Slicing is the workhorse of string manipulation. Run this quick FOR-NEXT loop for a display of its operation:

```
10 LET B$="ABCDE"
20 FOR X=1 TO 4
30 PRINT B$(X TO X)
40 NEXT X
```

We see that as *X* increases, the printed string changes:

when X is	the screen shows
1	A
2	AB
3	ABC
4	ABCD

This is the equivalent of *LEFT\$(B\$,X)* in common BASICs. Sinclair BASIC does not have this function.

Now change line 30 to:

```
30 PRINT B$(X TO )
```

Remember not to type in the spaces around *TO*—use SHIFT 4. RUN our program. Now we get:

when X is	the screen shows
1	ABCDE
2	BCDE
3	CDE
4	DE

This is equivalent to the *MID\$(B\$,X)* function found in other BASICs.

Try another change:

```
30 PRINT B$(X TO X+1)
```

Now we see:

when X is	the screen shows
1	AB
2	BC
3	CD
4	DE

This works like *MID\$(B\$,X,2)* in other BASICs.

Here's a tricky one. Change line 30 to read:

```
30 PRINT B$(LEN B$-X TO )
```

This produces:

when X is	the screen shows
1	DE
2	CDE
3	BCDE
4	ABCDE

This is the equivalent of *RIGHT\$(B\$,X)* in common BASICs.

SQ BEGINNERS' BASIC

Here's a quick program that gets the same result (input STEP using SHIFT E):

```
10 LET B$="ABCDE"
20 FOR X=4 TO 1 STEP -1
30 PRINT B$(X TO )
40 NEXT X
```

If you can visualize how and why these last two programs produce the same result, you have a good understanding of slicing.

Concatenation

Concatenation (pronounced kon-kat-e-NAY-shun) is a fancy word that means a combination, joining, linking, or chaining. We can concatenate, or link, strings by using the concatenation operator, the plus (+) sign. When it works on strings, it joins them. It doesn't perform arithmetic addition.

We can concatenate strings from string variables or characters enclosed in quotation marks. To see how this works, RUN this demo program:

```
10 LET F$="DIANA"
20 LET L$="JONES"
30 LET N$=L$+", "+F$
40 PRINT N$
```

Line 30 concatenates "JONES" in variable L\$, a string in quotation marks consisting of a comma and a space, and "DIANA" in variable F\$. It assigns the manufactured string to variable N\$. Line 40 prints it: JONES, DIANA

What's in a Name?

Let's assume we have a name stored in variable N\$, in a mailing list, for example. (In an actual program, we would probably have a series of names stored in an array.) We want to separate the first and last names into the variables F\$ (first name) and L\$ (last name).

Here's a routine to get the first name (remember, the computer ignores REM statements when executing a program):

```
10 LET N$="ANN DOYLE"
100 REM ROUTINE TO GET F$
110 FOR X=1 TO LEN N$
120 IF N$(X)=" " THEN GOTO 140
130 NEXT X
140 LET F$=N$(1 TO X-1)
150 PRINT F$
```

How does it work? Lines 110 and 130 set up a FOR-NEXT loop that sequentially examines each character of N\$. Line 120 is the "do" portion of the loop. It uses the slicing function to test each character one at a time for a space (" ") in N\$. When the space is found we have the first name.

Let's step through the loop and see what happens. On the first pass the index variable (counter), X, equals 1. The IF portion of line 120 looks at the first character, N\$(1), which is "A." It's not a space, so X is set to 2 for the second pass. The second character, N, is tested. On the fourth pass X=4 and the slicing function finds the

space it's looking for. Because the fourth (Xth) character is the space, we know that the first name is X-1 characters long. The THEN portion executes and jumps to line 140. Line 140 assigns the left X-1 (three) characters of N\$, which are ANN, to variable F\$.

Now let's find the last name. We'll use a different technique:

```
200 REM ROUTINE TO GET L$
210 FOR X=LEN N$ TO 1 STEP -1
220 IF N$(X)=" " THEN GOTO 240
230 NEXT X
240 LET L$=N$(X+1 TO )
250 PRINT L$
```

Here we step backwards from the right side of N\$, testing for a space. When the computer finds it, the THEN portion of line 220 executes and transfers program control to line 240. The slicer in line 240, (X+1 TO), deletes the left X characters from N\$ and returns the right side from character X+1 to the end of the string. The LET statement assigns the right side of N\$ to variable L\$.

Dimensioning Strings

Strings and string variables have size, or length. Regular string variables are the size of the alphanumeric information they contain. We can, however, set them to a fixed size, using a DIMension statement.

DIM statements define the number of characters in the string variable and initially fill it with that number of spaces (DIM is also used to DIMension arrays—a subject beyond our scope here). The statement DIM A\$(6), for example, creates a string of 6 spaces in variable A\$.

A\$ will remain 6 characters long no matter how many or how few characters we try to assign it. You can redimension a dimensioned string to a new size. But if you do, the computer will reinitialize it with spaces and erase its previous contents. Once dimensioned, a string cannot be undimensioned.

Chopping and Padding

What earthly good is a fixed-size string variable?

Suppose we have a field, or space, 16 characters wide for a name. We want to change our string so it has exactly 16 characters. How do we do it? Let's try it first with a long name, then with a short one:

```
10 LET N$="KORZENIOWSKI, STANISLOU"
10 LET N$="DOE, JANE"
```

Note that Stan's name has 23 characters and Jane's has 9.

We'll assign the result to variable X\$. It's easy—we'll just use the DIM statement to create a predefined string variable of 16 characters:

```
100 REM ROUTINE TO MAKE A 16-CHARACTER STRING
110 DIM X$(16)
120 LET X$=N$
```

SQ BEGINNERS' BASIC

Now we'll print the string from line 130, concatenating a period to show where the string ends. We'll print its length from line 140:

```
130 PRINT X$+"."
140 PRINT "LENGTH OF X$ IS ";LEN
N X$
```

Line 110 DIMensioned X\$ to 16 characters. When line 120 assigned N\$ into this 16-character X\$, Stan was chopped down to size—the computer chopped off the last seven characters of his name (the fancy word is “truncated”). When we assigned Jane’s 9-character name to the dimensioned variable, the computer added seven spaces to it (this is called padding).

Even if the previous contents of the dimensioned string were non-blank characters, as in Stan’s case, assigning a shorter string, like Jane’s name, erases the previous characters and pads the remainder of the string with spaces.

Experiment

Here’s one to try on your own. Give Jane and Stan middle names. Put their first names first in N\$ (for example, “Jane Carol Doe”). Now write a routine that puts each name in a 20-character field using this format: last name, comma, first initial, period, space, middle initial, period. For Jane’s name, the result should be: Doe, J. C. followed by 10 spaces.

Removing Trailing Spaces

When printing a string from a dimensioned string variable, you will often want to remove trailing spaces. (Have you ever received a computer-written letter that starts, “Dear Ms. Doe , You have just won...”?) Here’s a routine to get rid of those ugly spaces (use SHIFT T for the not-equal sign in line 120):

```
10 DIM T$(16)
20 PRINT "INPUT A TEST STRING"
30 INPUT T$
40 PRINT T$+"."
100 REM ROUTINE TO REMOVE TRAIL
ING SPACES
110 FOR X=LEN T$ TO 1 STEP -1
120 IF T$(X) <> " " THEN GOTO 140
130 NEXT X
140 LET A$=T$( TO X)
150 PRINT A$+"."
160 GOTO 20
```

The routine loops backward through T\$ with line 120 testing each character for a space. When it finds a non-space character (that is, the expression T\$(X) <> " " is true), the program goes to line 140. Here the string excluding the trailing spaces (that is, T\$(TO X)) is assigned to A\$. Lines 40 and 150 print the test string before and after removing the trailing spaces, concatenating a period on the end to show the length of the string.

Programming Tips

1. Function titles appear under most letter keys. Pressing the SHIFT and FUNCTION/ENTER keys will print a

reverse video F (function) cursor on the message line. This allows you to type the function title as you would a statement.

2. Parentheses around a function’s argument are optional unless the argument is a formula.

3. Functions, including slicing, have the last priority (12) in the computer’s order of operations. You’ll have to define concatenated string expressions by putting parentheses around them before slicing them. For example, you have to program the expression A\$+“,”+B\$ as (A\$+“,”+B\$) (slicer) to avoid slicing only B\$.

4. Use the single character slicer in input-checking routines to allow a yes/no input of either the first letter or the entire word. In the following example, YES goes back to the program start and NO stops. You can change these to suit your needs.

```
120 PRINT "ANSWER YES OR NO"
130 INPUT R$
140 IF R$(1)="Y" THEN GOTO 10
150 IF R$(1)="N" THEN STOP
```

5. A string variable, once dimensioned, can’t be undimensioned. But it can be redimensioned—and its contents reset to spaces. Previously dimensioned string variables are practically useless as regular variables. If you use dimensioned string variables more than once in a program (for chopping and padding, for example), see if you can redimension and recycle one you used before. You have a limited supply (26) of string variables, so unless you recycle, it’s easy in a complex program to run out of them.

6. The VAL function must be the first item in an arithmetic formula. If you can’t rewrite the formula to put VAL at the beginning, assign the VAL function to an arithmetic variable first. Then use the variable in the formula.

VAL, if used in a PRINT AT, PLOT or UNPLOT statement, must be the first (row) coordinate. To use it as the second (column) coordinate, assign it first to a variable. For example, change this line:

```
140 PRINT AT 12,VAL "C"
```

to these lines:

```
135 LET C=VAL "C"
140 PRINT AT 12,C
```

7. Your manual’s appendix shows the complete Timex Sinclair character set. Here’s a little program that lists them all:

```
10 REM PRINT CHARACTER SET
20 PRINT "CODE", "CHARACTER"
30 FOR X=0 TO 255
40 PRINT X,CHR$ X
50 NEXT X
```

Codes 67-111 and 122-127 will show as a “?” character. These codes aren’t used for characters.

TRANSLATION TABLE String Functions

No "standard" BASIC language exists — only dialects. This table summarizes the main differences between most common microcomputer BASIC dialects and Sinclair BASIC (used in ZX81s and Timex Sinclair 1000s). The table refers only to statements and operations used in or related to the accompanying article. It should help you translate programs written in common dialects into Sinclair BASIC. The accompanying article defines the italicized symbols.

<u>Example</u>	<u>What It Does or Is</u>	<u>How to Use It in Common BASIC</u>	<u>Sinclair BASIC</u>	<u>Translation</u>
LEFT\$ (\$, len)	Returns the left <i>len</i> characters of string \$.	Most dialects use.	Not available.	\$(TO <i>len</i>)
MID\$ (\$, pos, len)	Returns a sub-string of string \$, beginning with character number <i>pos</i> and having length of <i>len</i> characters.	Most dialects use.	Not available	\$(<i>pos</i> TO <i>pos</i> + <i>len</i>)
MID\$ (\$, pos) (<i>len omitted</i>)	Returns the entire sub-string right of position <i>pos</i> .	Most dialects use.	Not available	\$(<i>pos</i> TO)
RIGHT\$ (\$, len)	Returns the right <i>len</i> character of string \$.	Most dialects use.	Not available.	\$(LEN \$ - <i>len</i> TO)
ASC(\$)	Returns ASCII* code number of the first character of string \$. (* American Standard Code for Information Interchange)	Most use common code for punctuation, numbers, and uppercase letters but differ for graphics characters and cursor control codes.	Has own codes, unrelated to ASCII. No cursor control codes.	Use CODE \$
DIM \$(x)	Dimensions string variable \$ to <i>x</i> characters and fills with spaces.	A few dialects require dimensioning of strings, most don't allow.	Optional to predefine string size.	No changes needed. Use to simplify chopping and padding routines.
Redimensioning	Redimensions string variables (as above).	Most dialects don't allow.	No restriction.	No changes needed. See Tip 5 for use.
Parentheses	Enclose function's argument.	Most dialects require.	Optional on most functions. Slicers require (Tip 2).	Can usually be omitted. Best to include (for portability).

SYNCWARS— An Invasion

This game will help you improve your speed and accuracy with the ZX/TS keyboard while providing recreation.

Computerburg is being invaded by a squad of ten dread Synclons, who beam in, one at a time, from the hyperspace deep within your computer. Your job is to zap them as they appear on your screen. However, being clever little invaders, they each bring along a Syncloid Invisibility Robot (S.I.R.) which emits a field that hides the Synclon from view. Both the Synclon and the S.I.R. may appear anywhere on your viewscreen, but never in the same place.

At your disposal you have powerful lasers (the 38 keys on your keyboard, except SHIFT and SPACE/BREAK). By firing a laser corresponding to an invader's position you can destroy both types of invaders.

Press any key to start the invasion. The first Syncloid Invisibility Robot appears, surrounded by his field. You must destroy the robot to deactivate the field, thus making the Synclon invader visible. Next, you must zap the Synclon to score. If you miss the S.I.R., the field disturbance is enough to tingle the Synclon's aerals, giving away his position. If you're fast, and push the right laser in time (1/3 second) you can still hit the Synclon and score. Also, if you've destroyed the robot but miss the Synclon, you have a brief amount of time to correct your error and shoot again. Your shot is even valid if you miss the S.I.R. entirely but happen to hit the Synclon.

Destroying the robot slows the alien's transporter enough to give you about one and a half seconds to shoot the invaders, but this time decreases as subsequent Synclons beam in. To get the tenth team, you have only one second to hit the S.I.R. and then the Synclon. If you miss, or exceed the time limit, the Synclon vanishes again into your computer, to cause strange crashes and other nasty deeds. (Not really, but buy the premise and you'll enjoy the game more.)

To enter the program into a TS1000 (with 2K RAM) without RAM pack, see the instructions. The program will not fit into 1K ZX81s, but will run with an external RAM pack. If you are using an external RAM, see the 16K

modifications. You can simulate a 2K machine with your 16K ZX81 using POKE 16389,72, then NEW. The reason you should do this is that the screen-fill routine in line 1 speeds up considerably when memory isn't tight, since the display file is already padded out with spaces. The POKE followed by NEW resets your RAMTOP address to limit available RAM to 2K.

SYNCWARS With 2K RAM Instructions

1. Load Syntactic Sum Loader (SYNTAX newsletter, Aug. 82) into your 2K machine, and RUN, followed by NEW. (If you missed the Aug. SYNTAX, send us a self-addressed stamped envelope for a copy of *Syntactic Sum Loader*.—Ed.)

2. Type in the program as listed.

3. Check Syntactic Sum—it should equal 51036 (PRINT USR 18408). If it does not, check the program for errors. Make sure you run Syntactic Sum in the FAST mode.

4. Enter LET G\$="SYNCWARS " in immediate mode (note space at end). Start recording and enter GOTO 600. The program will SAVE to tape and stop with error 2/610. Rewind tape.

5. Reset your computer by pulling the plug or flipping the reset switch if you have one. This clears out the Syntactic Sum subroutine; there isn't enough memory space to run the game if any ML routines are present beyond RAMTOP.

6. LOAD "SYNCWARS " from tape (note the space at the end). When loaded, computer will stop with error 2/610. Rewind the tape.

7. In the immediate mode (no line numbers), type in the variables list. Do not CLEAR or RUN once you've started entering variables. Arrays B\$, C\$, E\$, and F\$ store the graphic pictures of the invaders and their destroyed counterparts; you may easily design your own figures on graph paper (each picture is five print positions tall and three positions wide) and then enter the strings line by line into the appropriate arrays.

8. Again save the program (this time with variables loaded) to tape, overwriting the first version. Program starts when saved.

```

1 REM Y. # : 4NOT $TAB RND TAB
RNDTAN
10 PRINT AT R,T;" ";G$,,,,"BY
F.NACHBAUR"
25 POKE 16418,0
30 PRINT AT M,0;"PRESS P TO PL
AY"
40 PAUSE U
50 RAND
100 LET S=0
110 FOR C=P TO T
120 CLS
130 LET D=INT (RND*U)
140 LET E=INT (D/T)
150 LET G=(D-T*E)*Q+E
160 LET E=E*N-P
180 LET H=INT (RND*U)
185 IF H=D THEN GOTO U+V
190 LET I=INT (H/T)
210 LET K=(H-T*I)*Q+I
215 LET I=I*N-P
220 PRINT USR 16514;AT M,0;H$
260 FOR A=P TO R
264 PRINT AT E+A,G;C$(A)
268 NEXT A
270 PAUSE U-Q*C
290 IF INKEY$="" THEN GOTO U*R
300 IF INKEY$=A$(H+P) THEN GOTO
U*T
310 IF INKEY$<>A$(D+P) THEN GOT
O L-T
320 CLS
330 FOR A=P TO R
335 PRINT AT E+A,G;E$(A);AT I+A
,K;B$(A)
340 NEXT A
350 PAUSE U-Q*C
370 IF INKEY$="" THEN GOTO U*R
375 IF INKEY$<>A$(H+P) THEN GOT
O L-T
400 FOR A=P TO R
404 PRINT AT I+A,K;F$(A)
408 NEXT A
410 LET S=S+P
420 PRINT AT M,0;D$;S
430 PAUSE U
440 GOTO L
450 CLS
460 PRINT AT T,T;"TOO SLOW "
465 PAUSE U
470 GOTO L
490 PRINT AT I+P,K;"*#*"
494 PAUSE U
495 IF INKEY$=A$(H+P) THEN GOTO
U*T
500 NEXT C
550 PRINT AT T,T;"GAME OVER",,,
D$;S;H$;
560 IF S=T THEN PRINT "TOP SCOR
ER"
570 GOTO M
600 SAVE G$
610 GOTO T
SYNTACTIC SUM: 51036, 8K ROM

```

Notes for 2K Listing

- 1 Graphics on T,H,5,Y
- 10 Inverse space
- 30 "P,R,E,S,S,spc,P,token TO ,P,L,A,Y"
- 220 Screen fill
- 460 Token "SLOW"
- 490 Second character is inverse ""
- 560 Inverse video

Variables

(Enter in immediate mode)

```

LET L=500
LET M=23
LET N=6

```

```

LET O=0
LET P=1
LET Q=3
LET R=5
LET T=10
LET U=38
LET V=90
LET W=40000
LET A$="1234567890QWERTYUIOPASDF
GHJKL ZXCVBNM."

```

Note space between L and Z.

```

LET D$="SCORE="
LET G$="SYNCWARS "
LET H$=""
LET A$(30)=CHR$ 118

```

Note space at end.
Space.

"Enter" character (replaces space in A\$).

```

DIM B$(5,3)
DIM C$(5,3)
DIM E$(5,3)
DIM F$(5,3)
LET B$(1)="><"
LET B$(2)="E_R"

```

Synclon array.

Synclon array.

Broken Synclon.

Destroyed Synclon.

Greater than spc less than.

Graphic E, inverse spc, graphic R.

```
LET B$(3)="_F_"
```

Inverse spc, graphic F, inverse spc.

```
LET B$(4)="W6Q"
```

Graphic W, graphic 6, graphic Q.

```
LET B$(5)="Q7W"
```

Graphic Q, graphic 7, graphic W.

```
LET C$(1)=B$(1)
LET C$(2)=B$(2)
LET C$(3)="_7_"

```

Inverse spc, graphic 7, inverse spc.

```
LET C$(4)="Q_W"
```

Graphic Q, inverse spc, graphic W.

```
LET C$(5)="E R"
```

Graphic E, normal spc, graphic R.

```
LET E$(1)="/"
LET E$(2)("<<")
LET E$(3)="TE*"

```

Spc/spc

Less than spc less than.

Graphic T, graphic E, normal *

```

LET E$(4)=C$(4)
LET E$(5)=C$(5)
LET F$(1)="GOT"
LET F$(2)=""
LET F$(3)=""
LET F$(4)=F$(2)
LET F$(5)="M"E"

```

Second * is inverse.

SYNCWARS For 16K Machines

With a 16K RAM pack, you may POKE 16389,72 and POKE 16388,24 then NEW to give you enough room for Syntactic Sum plus the program and variables. The PRINT USR address in this case is 18432, and you don't have to dump the ML routine before loading the variables and starting the program going.

If you'd prefer not to POKE RAMTOP, but wish to use your 16K machine, enter the program as written and add the following lines:


```

1 REM SYNCWARS
220 FOR A=O TO M-P
230 PRINT "....."
240 NEXT A
500 FOR A=P TO RND*SQR W
510 NEXT A
520 NEXT C
NEW SYNTACTIC SUM = 53311

```

Replaces old line
1 ML screen fill

Replaces old line
500

```

1 REM SYNCWARS
10 PRINT AT R,T;"■";G$;,,,,"BY
F.NACHBAUR"
25 POKE 16418,0
30 PRINT AT M,0;"PRESS P TO PL
AY"
40 PAUSE W
50 RAND
100 LET S=0
110 FOR C=P TO T
120 CLS
130 LET D=INT (RND*U)
140 LET E=INT (D/T)
150 LET G=(D-T*E)*Q+E
160 LET E=E*N-P
180 LET H=INT (RND*U)
185 IF H=D THEN GOTO U+V
190 LET I=INT (H/T)
210 LET K=(H-T*I)*Q+I
215 LET I=I*N-P
220 FOR A=O TO M-P
230 PRINT "....."
240 NEXT A
260 FOR A=P TO R
264 PRINT AT E+A,G;C$(A)
268 NEXT A
270 PAUSE V-Q*C
290 IF INKEY$="" THEN GOTO V*R
300 IF INKEY$=A$(H+P) THEN GOTC
U*T
310 IF INKEY$(<>)A$(D+P) THEN GOT
O L-T
320 CLS
330 FOR A=P TO R
335 PRINT AT E+A,G;E$(A);AT I+A
,K;B$(A)
340 NEXT A
350 PAUSE V-Q*C
370 IF INKEY$="" THEN GOTO V*R
375 IF INKEY$(<>)A$(H+P) THEN GOT
O L-T
400 FOR A=P TO R
404 PRINT AT I+A,K;F$(A)
408 NEXT A
410 LET S=S+P
420 PRINT AT M,0;D$;S
430 PAUSE V
440 GOTO L
450 CLS
460 PRINT AT T,T;"TOO SLOW "
465 PAUSE U
470 GOTO L
490 PRINT AT I+P,K;"■"
494 PAUSE U
496 IF INKEY$=A$(H+P) THEN GOTO
U*T
500 FOR A=P TO RND*SQR W
510 NEXT A
530 NEXT C
550 PRINT AT T,T;"GAME OVER",,
D$;S;H$;
560 IF S=T THEN PRINT "STOE SC
RE"
570 GOTO M
600 SAVE G$
610 GOTO T
SYNTACTIC SUM: 53311, 8K ROM

```

This modification randomizes the length of time between successive invader pairs, making the game a little less predictable. The routine in lines 220-240, while taking more memory than the MC screen fill (and therefore unusable in the 2K version), is faster and allows the next Syncloid to appear almost immediately, or delay for a while, as determined by the dummy loop in lines 500-510.

Running the Program

The program starts automatically when loading from tape. When playing the game be careful not to press the SPACE/BREAK key; to do so will break operation. CONTINUE will not work to restart the program, making it cheat-proof. If you hit BREAK by mistake, you can start the program again by entering GOTO 0 (or GOTO O or GOTO P).

ZX80s

Since the program runs in the FAST mode, it will run in ZX80s with 8K ROM, even without video upgrade mods. Minimum RAM requirement is 2K.

1K Machines

Finally, here's a listing of the game which will run in 1K ZX81s and ZX80s with 8K ROM update. As with the 2K version, you must load Syntactic Sum (if used) and then type in the program; check Syntactic Sum, and when

Available from SYNTAX...

For computing beginners —

Crash Course in Microcomputers\$19.95
Covers hardware, machine language and applications. Reviewed in SYNTAX, Oct. 1981. Add \$1.50 shipping.

ZX80 Pocket Book\$10.95
Includes ZX81 supplement. Covers Sinclair BASIC, data and program listings. Add \$1.50 postage.

For advanced hardware/software users —

Zilog's Z80-Z80A CPU Technical Manual\$7.88

Zilog's Assembly Language Programming\$15.75

Experiments in Artificial Intelligence\$8.95
Add \$1.50 postage.

SYNTAX back issues available, \$4 each.
Call or write for our group subscription discounts.

SYNTAX • RD 2 Box 457 • Harvard, MA 01451
617 / 456-3661

correct save the program to tape. Then reset the machine, load from tape, and enter the variables. Type GOTO 0 to start the game. Don't forget to save the program with variables, overwriting the first SAVE.

This version is more difficult, since you don't have a second chance to get the invaders. Also, the display is considerably smaller and the invaders' positions are not staggered to match the layout of the keyboard. Note that if the invaders appear at any of the last three positions in the bottom row, you must hit "." to score.

Notes for 1K Version

124 20 periods.

250 Graphics on R,E; E,R.

320 Graphics on 5,Y; 2 inverse "."; Graphics on T,Y.

Variables

LET L=500

LET P=1

LET Q=2

LET R=4

LET T=10

LET V=90

LET A\$="1234567890QWERTYUIOPASDF

GHJKL ZXCVBNM..." (LEN A\$ should be 40)

LET A\$(30)=CHR\$ 118

To start, enter GOTO 0 or GOTO P. In this version, if you hit the SPACE/BREAK key by mistake, it's ok to type CONTInue and keep going. If you type RUN by mistake, reload the variables list and start with GOTO. You may prefer to run this version in SLOW mode. **SO**

```

5 PRINT AT R,R;"SYNCWARS"
10 PAUSE V
100 LET S=P-P
110 FOR C=P TO T
115 CLS
122 FOR E=Q TO T-P
124 PRINT "....."
126 NEXT E
130 LET E=INT (RND*R)
140 LET G=INT (RND*T)
150 LET D=T*E+G+P
160 LET E=E*Q
170 LET I=INT (RND*R)
190 LET K=INT (RND*T)
200 LET H=T*I+K+P
205 IF H=D THEN GOTO U+V
210 LET I=I*Q
250 PRINT AT E,G*Q;"██";TAB G*Q
270 PAUSE U
280 IF INKEY$=A$(H) THEN GOTO L
290 IF INKEY$<>A$(D) THEN GOTO
300 CLS
320 PRINT AT E,G*Q;"</" ;TAB G*Q
325 PAUSE U
330 IF INKEY$<>A$(H) THEN GOTO
410 LET S=S+P
420 PRINT AT I,K*Q;"*/";TAB K*Q
495 PAUSE U/Q
500 NEXT C
510 PRINT AT T,P;"YOU GOT ";S;"
SYNCLONS"
SYNTACTIC SUM: 29905, 8K ROM

```

IC INTERCOMPUTER INC. TM

presents

A Full Line Of Quality SOFTWARE For Your TIMEX 1000 Or SINCLAIR ZX81

IC's new line of software includes action-packed, challenging programs which put the user in the pilot's seat of an air force bomber; into the casinos of Las Vegas; behind the controls of a nuclear missile launcher; alongside the President as his bodyguard; in a lost spaceship, far from earth; in command of a submarine destroyer, and challenge him or her to other mind and strategy games.

**DEALERS AND DISTRIBUTORS
WELCOME**

WE ACCEPT CALL ORDERS
WITH VISA OR MASTERCARD

Write or Mail to:

INTERCOMPUTER, INC.

P. O. Box 90, Prudential Center

Boston, Mass. 02199 (617) 437-1190

order form

FG #	TITLE	QTY	U. S. \$	TOTAL
1001	Return From Space		10.95	
1002	Missile Launcher		10.95	
1003	Jeopardy		10.95	
1004	Vegas		10.95	
1005	Demolisher		10.95	
1006	Air Attack		10.95	
1007	Guard the President		10.95	
1008	Combo PAK I		14.50	
1009	Combo PAK II		14.50	
1010	Combo PAK III		14.50	
1011	Submarine		10.95	
1012	Combo PAK IV		14.50	
P001	Memopak 16 K RAM		59.95	
P002	Memopak 32 K RAM		109.95	
P003	Memopak 64 K RAM		179.95	

All Programs require 16K RAM

*Defective tapes will be replaced, if returned within 3 days.

*Please allow 2 weeks for delivery.

*Your order may be sent in more than one shipment.

Sub Total

*5%

Mass. Tax

TOTAL

*Mass. Residents Only

☐ Check enclosed

☐ Charge my credit card below

Name

Company

Street

City

Signature

☐ VISA ☐ MasterCard

Card Number

Expiration Date

SOFTSYNC, INC.

**THE WORLDS BEST PROGRAMS
THE WORLDS BEST PROGRAMMERS**

**JOIN THE FAMILY. CALL OR WRITE FOR A FREE TS1000/ZX81 CATALOG
AND PROGRAM LISTING AND OR DETAILS ON PROGRAM DISTRIBUTION**

14 E. 34th St. NEW YORK, NEW YORK, 10016 212-685-2080

Geography Review—U.S. States Quiz Program

I recently adapted an idea for a program from Tab's *The A to Z Book of Computer Games*. The result is a states quiz that reviews the geography of the U.S. It is a good teaching tool for children and a good review for adults.

The program requires the 8K ROM with 16K RAM and works best in the FAST mode.

Your objective is to match a state (the program picks one at random) to a list of 9 regions and 1 state capitol in the United States. If you fail on your first try, the program gives you a second chance. If you fail again, you get another try later on in the quiz.

The program keeps giving you states to match until you get all 50 right. When you answer all 50 states correctly the program tells you how many guesses you took. The first guess is free: the computer only counts a wrong answer if you miss it twice. This helps not to discourage children from completing the quiz. If you want to refer to the quiz instructions at anytime, just push key I (for instructions) and they will appear on the screen. To see your score at anytime, push key S. To quit, push Q.

After ENTERing the program, hit RUN and ENTER. The program is SAVED under the title "USA".

U.S. States Quiz uses the string slicing capabilities of the Sinclair 8K ROM as the method of getting information out about the 50 states. To speed up the program, the most frequently used routines are at the beginning. The program also makes use of the INKEY\$ function instead of INPUT followed by ENTER. This reduces the number of keys you must press.

Listing 1 describes the variables used and their functions. Instructions for *U.S. States Quiz* are contained in lines 8010-8090. To exit from the program, just push BREAK.

Listing 1

- A — How many times (0 or 1) you missed the question.
- B — How many questions you answered correctly.
- C — How many questions you answered incorrectly.
- D — The number of states you have been asked to match.

- E — Does the player have a name? (0 if no)
- F — Random number (1 to 50) picks which state the question is about.
- G — Counts the states to see if all have been answered correctly.
- A\$(F, 1 to 16)—State data. The first element of the string is the location (0 to 9), the second is if it has been guessed (1) or not (0), the remainder of the string names the state.
- B\$ — Player's name.
- C\$ — Key on the keyboard that has been pushed.
- D\$ — Name of the location picked.
- E\$ — Name of the correct location.

SQ

```

10 REM PROGRAM NAME IS USA
20 CLS
30 GOSUB 9000
40 PRINT "WANT INSTRUCTIONS (Y
OR N)"
50 PAUSE 40000
60 IF INKEY$="N" THEN GOTO 80
70 GOTO 8000
80 CLS
90 PRINT "WHAT IS YOUR NAME?"
100 INPUT B$
110 LET E=1
120 LET F=INT (RND*50)+1
130 IF A$(F,2) <> "0" THEN GOTO 1
20
140 GOTO 1000
150 PAUSE 40000
160 LET C$=INKEY$
170 CLS
180 IF C$="I" THEN GOTO 8000
190 IF C$="S" THEN GOTO 7000
200 IF C$="Q" THEN GOTO 6000
210 REM ERROR TRAP
220 IF C$="0" OR C$="1" OR C$="
2" OR C$="3" OR C$="4" OR C$="5"
OR C$="6" OR C$="7" OR C$="8" O
R C$="9" THEN GOTO 250
230 GOTO 1000
240 REM LOCATION GUESSED
250 IF C$="0" THEN LET D$="REMO
TE"
260 IF C$="1" THEN LET D$="PACI
FIC"
270 IF C$="2" THEN LET D$="MEXI
CO"
280 IF C$="3" THEN LET D$="GULF

```

Continued Next Page


```

290 IF C$="4" THEN LET D$="ALAN
TIC"
300 IF C$="5" THEN LET D$="GREA
T LAKES"
310 IF C$="6" THEN LET D$="CANA
DA"
320 IF C$="7" THEN LET D$="CHAR
LESTON"
330 IF C$="8" THEN LET D$="MISS
. RIVER"
340 IF C$="9" THEN LET D$="WEST
ERN"
350 IF A$(F,1)=C$ THEN GOTO 200
0
360 IF A<>0 THEN GOTO 470
370 PRINT "SORRY,";B$
380 PRINT D$;" IS NOT THE MATCH
TO"
390 PRINT A$(F,3 TO 16)
400 PRINT "YOU GET ONE MORE C
HANCE TO MATCH"
410 PRINT A$(F,3 TO 16)
420 PRINT "TO THE RIGHT LOCATIO
N."
430 PRINT "PUSH ANY KEY (EXCE
PT SPACE) TO TRY AGAIN."
440 LET A=1
450 PAUSE 40000
460 GOTO 1000
470 PRINT "SORRY ";B$
480 PRINT "YOU MISSED TWICE"
490 PRINT "E$
500 PRINT "WAS THE RIGHT ANSWER
FOR"
510 PRINT A$(F,3 TO 16)
520 PRINT "PUSH ANY KEY (EX
CEPT SPACE) FOR THE NEXT STATE"
530 LET C=C+1
540 LET D=D+1
550 LET A=0
560 PAUSE 40000
570 GOTO 120
990 REM INSTRUCTIONS
1000 CLS
1010 PRINT "HELLO ";B$
1020 PRINT AT 0,30;D
1030 PRINT "WHICH LOCATION WIL
L MATCH"
1040 PRINT A$(F,3 TO 16)
1050 PRINT "*****"
*****
1060 PRINT AT 6,11;"LOCATIONS"
1070 PRINT "0) REMOTE (2
STATES MATCH"
1080 PRINT "1) PACIFIC (3 ST
ATES MATCH)"
1090 PRINT "2) MEXICO (3 ST
ATES MATCH)"
1100 PRINT "3) GULF (4 ST
ATES MATCH)"
1110 PRINT "4) ALANTIC (13 S
TATES MATCH)"
1120 PRINT "5) GREAT LAKES (7 ST
ATES MATCH)"
1130 PRINT "6) CANADA (4 ST
ATES MATCH)"
1140 PRINT "7) CHARLESTON (1 ST
ATE MATCHES)"
1150 PRINT "8) MISS. RIVER (5 ST
ATES MATCH)"
1160 PRINT "9) WESTERN (8 ST
ATES MATCH)"
1170 PRINT "PUSH ""I"" FOR T
HE INSTRUCTIONS"
1180 IF A$(F,1)="0" THEN LET E$=
"REMOTE"
1190 IF A$(F,1)="1" THEN LET E$=
"PACIFIC"
1200 IF A$(F,1)="2" THEN LET E$=
"MEXICO"
1210 IF A$(F,1)="3" THEN LET E$=
"GULF"
1220 IF A$(F,1)="4" THEN LET E$=
"ALANTIC"
1230 IF A$(F,1)="5" THEN LET E$=
"GREAT LAKES"

```

```

1240 IF A$(F,1)="6" THEN LET E$=
"CANADA"
1250 IF A$(F,1)="7" THEN LET E$=
"CHARLESTON"
1260 IF A$(F,1)="8" THEN LET E$=
"MISS. RIVER"
1270 IF A$(F,1)="9" THEN LET E$=
"WESTERN"
1280 GOTO 150
1990 REM RIGHT ANSWER ROUTINE
2000 PRINT "WOW,";B$
2010 PRINT "D$;" MATCHES ";A$(F
,3 TO 16)
2020 PRINT "YOU PICKED THE RIGHT
ANSWER."
2030 PRINT "PUSH ANY KEY (EX
CEPT SPACE) FOR THE NEXT STATE."
2040 PAUSE 40000
2050 LET A$(F,2)="1"
2060 LET A=0
2070 LET B=B+1
2080 LET D=D+1
2090 FOR G=1 TO 50
2100 IF A$(G,2)="0" THEN GOTO 12
0
2110 NEXT G
2120 REM ALL 50 STATES MATCHED C
ORRECTLY
2130 CLS
2140 PRINT "VERY GOOD ";B$
2150 PRINT "YOU HAVE MATCHED ALL
50 STATES"
2160 PRINT "CORECTLY."
2170 PRINT "*****THESE ARE YO
UR SCORES*****"
2180 PRINT "STATES ATTEMPTED
TO MATCH ";D-1
2190 PRINT "STATES MATCHED COR
RECTLY ";B
2200 PRINT "STATES MATCHED INC
ORRECTLY ";C
2210 PRINT "*****"
*****
2220 PRINT "TO PLAY AGAIN PU
SH ""Y""
2230 PRINT "TO END THE QUIZ PU
SH ""Z""
2240 PAUSE 40000
2250 CLS
2260 IF INKEY$="Y" THEN RUN
2270 IF INKEY$="Z" THEN GOTO 229
0
2280 GOTO 2140
2290 PRINT AT 10,5;"GOODBYE ";B$
2300 STOP
5990 REM STOPPING THE QUIZ
6000 CLS
6010 PRINT B$
6020 PRINT "TO START AGAIN FRO
M THE BEGINNING PUSH ""X
""
6030 PRINT "TO CONTINUE THE QU
IZ PUSH ""Y""
6040 PRINT "TO END THE QUIZ PU
SH ""Z""
6050 PAUSE 40000
6060 IF INKEY$="X" THEN RUN
6070 IF INKEY$="Y" THEN GOTO 100
0
6080 IF INKEY$="Z" THEN GOTO 610
0
6090 GOTO 6000
6100 CLS
6110 PRINT AT 10,5;"GOODBYE ";B$
6120 PAUSE 520
6130 STOP
6990 REM SCORE
7000 CLS
7010 PRINT B$;" THESE ARE YOUR
SCORES-"
7020 PRINT "STATES ATTEMPTED
TO MATCH ";D-1
7030 PRINT "STATES MATCHED COR
RECTLY ";B

```

Continued Next Page

```

7040 PRINT "STATES MATCHED INCORRECTLY";C
7050 PRINT "PUSH ANY KEY (EXCEPT SPACE) TO CONTINUE."
7060 PAUSE 40000
7070 GOTO 1000
7990 REM INSTRUCTIONS
8000 CLS
8010 PRINT "THIS QUIZ WILL TEST TO SEE IF YOU CAN MATCH A STATE WITH ITS LOCATION (REGION OR STATE CAPITOL) IN THE U.S."
8020 PRINT "AFTER YOU SEE THE FIRST STATE ON THE SCREEN*****"
8030 PRINT "PUSH 0-9 TO MATCH THE STATE SHOWN TO ONE OF THE LOCATIONS SHOWN."
8040 PRINT "PUSH ""I"" TO SEE THE INSTRUCTIONS AGAIN."
8050 PRINT "PUSH ""S"" TO SEE YOUR SCORE."
8060 PRINT "PUSH ""Q"" TO END THE QUIZ."
8070 PRINT "THE LOCATIONS ARE LISTED IN THE PRIORITY TO BE ANSWERED (EXAMPLE-IF A STATE BORDERS BOTH CANADA AND THE GREAT LAKES, GREAT LAKES WILL BE THE CORRECT ANSWER BECAUSE IT IS HIGHER UP ON THE LIST OF LOCATIONS.)"
8080 PRINT "*****"
8090 PRINT "NOW PUSH ENTER TO BEGIN."
8100 PAUSE 40000
8110 IF E=0 THEN GOTO 80
8120 IF E<>0 THEN GOTO 1000
8990 REM LOAD STATES*****THE FIRST ELEMENT OF THE STRING IS THE LOCATION, THE SECOND IS IF IT HAS NOT BEEN GUESSED (0) OR IF IT HAS BEEN GUESSED (1), THE REMAINDER OF THE STRING IS THE NAME OF THE STATE.
9000 DIM A$(50,16)
9010 LET A$(1)="30ALABAMA"
9020 LET A$(2)="00ALASKA"
9030 LET A$(3)="20ARIZONA"
9040 LET A$(4)="80ARKANSAS"
9050 LET A$(5)="10CALIFORNIA"
9060 LET A$(6)="90COLORADO"
9070 LET A$(7)="40CONNECTICUT"
9080 LET A$(8)="40DELAWARE"
9090 LET A$(9)="30FLORIDA"
9100 LET A$(10)="40GEORGIA"
9110 LET A$(11)="00HAWAII"
9120 LET A$(12)="60IDAHO"
9130 LET A$(13)="50ILLINOIS"
9140 LET A$(14)="50INDIANA"
9150 LET A$(15)="80IOWA"
9160 LET A$(16)="90KANSAS"
9170 LET A$(17)="80KENTUCKY"
9180 LET A$(18)="30LOUISIANA"
9190 LET A$(19)="40MAINE"
9200 LET A$(20)="40MARYLAND"
9210 LET A$(21)="40MASSACHUSETTS"
9220 LET A$(22)="50MICHIGAN"
9230 LET A$(23)="50MINNESOTA"
9240 LET A$(24)="30MISSISSIPPI"
9250 LET A$(25)="80MISSOURI"
9260 LET A$(26)="60MONTANA"
9270 LET A$(27)="90NEBRASKA"
9280 LET A$(28)="90NEVADA"
9290 LET A$(29)="40NEW HAMPSHIRE"
9300 LET A$(30)="40NEW JERSEY"
9310 LET A$(31)="20NEW MEXICO"
9320 LET A$(32)="40NEW YORK"
9330 LET A$(33)="40NORTH CAROLINA"
9340 LET A$(34)="60NORTH DAKOTA"
9350 LET A$(35)="50OHIO"

```

```

9360 LET A$(36)="90OKLAHOMA"
9370 LET A$(37)="10OREGON"
9380 LET A$(38)="50PENNSYLVANIA"
9390 LET A$(39)="40RHODE ISLAND"
9400 LET A$(40)="40SOUTH CAROLINA"
9410 LET A$(41)="90SOUTH DAKOTA"
9420 LET A$(42)="80TENNESSEE"
9430 LET A$(43)="20TEXAS"
9440 LET A$(44)="90UTAH"
9450 LET A$(45)="60VERMONT"
9460 LET A$(46)="40VIRGINIA"
9470 LET A$(47)="10WASHINGTON"
9480 LET A$(48)="70WEST VIRGINIA"
9490 LET A$(49)="50WISCONSIN"
9500 LET A$(50)="90WYOMING"
9510 RAND
9520 REM HOW MANY TIMES YOU HAVE MISSED THE QUESTION
9530 LET A=0
9540 REM HOW MANY RIGHT
9550 LET B=0
9560 REM HOW MANY WRONG
9570 LET C=0
9580 REM HOW MANY STATES HAS THE PLAYER BEEN ASKED TO MATCH
9590 LET D=1
9600 REM NO NAME YET
9610 LET E=0
9620 RETURN
SYNTACTIC SUM: 20220, 8K ROM

```

We will pay CASH

for non-exclusive rights to
manufacture and sell your
Timex/Sinclair educational and
entertainment programs.

DALLAS DEVELOPMENT SYSTEMS
7410 Stillwater
Garland, Texas 75042
(214) 238-1776

POKEing Directly To The Display File

In SYNTAX newsletter (Nov. 81) a neat games program appeared called "MONXZER" (revised version Winter 82 issue of SQ). But the graphics are sooooo sloooooow and take up sooooo much memory. How can we avoid this tremendous memory consumption and delay with graphics?

First, it isn't necessary to directly use PRINT or PLOT statements. You can write directly into the display file, the place in memory where the computer stores the information to be shown on the screen.

Try this:

```
POKE PEEK 16396 + 256 * PEEK 16397 + 114, 23
```

An asterisk should appear near top center of your screen. (I will explain why later). The display file's memory seems to be set up as follows:

Enter (Code 118)	32 bytes of text
Enter	32 bytes of text
Enter	32 bytes of text
Enter	32 bytes of text, etc.

Using the enter symbol as the first byte of each line seems to key the video—without it the screen goes crazy. So you shouldn't write (POKE) into these locations. To count out *where* to POKE a variable, add 33 for each line past the first ENTER (code 118) plus the number of characters (columns) over to be displayed.

You may find the location of the first "Enter" by looking into the system variable D_FILE (memory pair 16396 and 16397). Thus, we POKE the asterisk (symbol 23 in the character set table) into the third row 3×33 and the 15th column. The two PEEKs tell the computer where to look for the start of the display file. Adding $3 \times 33 + 15$, or 114, tells it to move to the third row, 15th column. So the location to POKE the asterisk symbol to is given by:

```
PEEK(16396) + 256 * PEEK(16397) + 3 * 33 + 15
```

I don't recommend using PEEKs and POKEs for speed and certainly not for memory conservation. However, they do demonstrate how to get into the display file. Now

we may use (you guessed it) machine coding! It is *much* faster, and memory economical. Consider the following program:

LD HL, 16396	; first location for D_FILE
LD BC, 81	; first position 2nd row, 15th col.
LD A, 23	; 23 is code for symbol *
ADD HL, BC	; HL points to 2nd row, 15th col.
LD (HL), A	; put in *
LD BC, 33	; displacement for 1 row.
ADD HL, BC	; next row
LD (HL), A	; put in next *
ADD HL, BC	; next row
LD (HL), A	; put in next *
ADD HL, BC	; next row
LD (HL), A	; put in next *
RET	; go back to main program

You can load this routine into a REM statement at line 1. (See also *Machine Code Programs—Where and How to Load Them* by William Wentz, SQ, Winter 1982.) Enter 1 REM AAAAAAA... (three rows are enough). Then POKE the following values in numerical order. For example, type POKE 16520, 42 then press ENTER or NEW-LINE. Then type POKE 16521, 12 and press ENTER and so on. Don't put any line numbers before the POKEs.

LD HL	16520, 42	16531, 33
	16521, 12	16532, 0
	16522, 64	16533, 9
LD BC	16523, 1	16534, 119
	16524, 81	16535, 9
	16525, 0	16536, 119
LD A	16526, 62	16537, 9
	16527, 23	16538, 119
	16528, 9	16539, 201
LD (HL) A	16529, 119	RET
LD BC	16530, 1	

ADD HL, BC

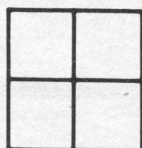
Then enter this line:

```
10 LET A =USR 16520
```

Four asterisks will appear *very* fast (even in SLOW mode) when you RUN this two-line program.

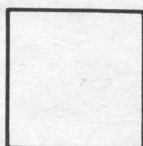
I recommend learning how to use machine code for these kinds of problems. Check out *TRS-80 Assembly Language Programming* by William Barden for a discussion of various techniques (useful for *any* assembly or machine language) and *Z80 Users Manual* by Joseph J. Carr for a more detailed discussion of the instructions. (Look out! His codes in binary aren't always correct! These are contained in the back of the TRS-80 text.)

Lastly, we may look at the pixels and how to address them in machine code. Each character block on the screen is split up into four segments. If we label bits from least to most significant beginning with one:

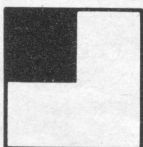


87654321
00000000

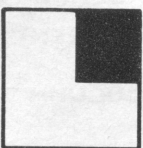
We see that all zeros are blank (white), a 1 in bit one turns on (blackens) the upper left hand corner, the second bit the upper right, and the third the lower left.



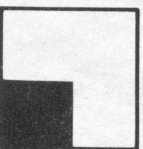
= 00000000



= 00000001

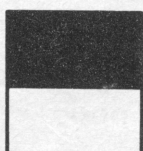


= 00000010



= 00000100

We can form graphics symbols by darkening combinations of pixels:

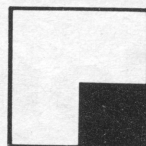


= 00000011

If you wondered what happened to the lower right hand pixel, it's coming right up. We may form an *inverse image* by putting a 1 in bit 8:



= 00000111



= 10000111

We thus see that the addressing used by the PLOT instruction is somewhat more complicated than at first glance. Also, if you know what symbols go where (not doing mathematical graphs), you just need to store them in a table and use a machine-coded segment to place them in position.

In closing, I'd like to indicate that 1) machine code isn't complicated; 2) it is tedious; 3) it is *very* fast; 4) it is *very* memory efficient; and 5) some programming problems involving graphics display are best solved with machine coding.

(Ed. note: This program requires 4K RAM or more to run.)

SQ



GENERAL SYSTEMS CONSULTING

2312 Rolling Rock Drive
Conley, Georgia 30027

SINCLAIR ZX81 and

TIMEX SINCLAIR 1000 SOFTWARE

16K minimum configuration

Designed to help monitor your finances.

- | | |
|---|------|
| 1. Amortizations | 9.95 |
| 2. Bar Charts | 9.95 |
| 3. Annuity Evaluation | 9.95 |
| 4. File Manager | 9.95 |
| 5. Bank Statement Balancer | 9.95 |
| 6. Checkbook Simulator | 9.95 |
| 7. Depreciation Straight Line | 9.95 |
| 8. Depreciation Declining Balance | 9.95 |
| 9. Depreciation (ACRS) | 9.95 |
| 10. Diet Plan | 9.95 |
| 11. Home Budget | 9.95 |
| 12. Home Inventory | 9.95 |
| 13. Home Payables | 9.95 |
| 14. Home Equity Evaluation | 9.95 |
| 15. Real Estate Investing | 9.95 |
| 16. Savings/Investments Analysis | 9.95 |
| 17. IRS 1040 (Long Form) | 9.95 |
| 18. IRS 1040A (Short Form) | 9.95 |
| 19. Income Tax Projections | 9.95 |

Circle selections and fill out form below:

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19

Number of items selected _____ @ 9.95

Postage/Handling 1.50

Total: _____

NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

Simultaneous Linear Equations and Matrix Inversion

Solves simultaneous linear equations or inverts an $N \times N$ matrix by forward and backward row reduction. A delight for amateur mathematicians.

Given a simple system of linear equations, for example, if $X + 2Y = 5$ and $X = 2$, find X , your favorite junior high school whiz would no doubt arrive at the correct answer before you could even find the tape to load your computer. But as the size of the problem grows to involve more than about three variables, the solution can become quite tedious and prone to error. There is also the chance that after much arithmetic you will find that no unique solution to the problem exists. So if you have say a 10 by 10, you can bet that your ZX/TS will beat any pencil pusher several times over.

Matrix algebra is a little more complex, but the row reduction method is essentially the same as for simultaneous equations and it is convenient to include both in one program. I'll not even begin to explain why one would want to invert a matrix, but matrix inversion is rather akin to taking the joint reciprocal of a specific array of numbers. If you multiply the inverted matrix by the original matrix, you see the operation is similar to the simple $1/X \times X = 1$.

To use this program, just type it in and RUN. It is fully prompted, so just enter the correct responses. The program starts by asking for the type and size of problem, sets appropriate dimensions and prompts the necessary input. I calculate that 16K RAM will handle at least a 25×35 matrix or about 50 simultaneous linear equations. During input the coefficient matrix is augmented as necessary by either the output matrix for equations, or an identity matrix for inversion. You can make corrections after entering all data. To exit, just answer N after your problem is solved.

Forward row reduction proceeds by stepping across the columns and down each row (along the diagonal), selecting at each step a sub-row suitable for reducing each

```

10 PRINT "THIS PROGRAM SOLVES"
20 PRINT "SIMULTANEOUS LINEAR"
30 PRINT "EQUATIONS (E) OR"
40 PRINT "INVERTS AN N*N MATRI
X (I)"
50 PRINT
60 PRINT "WHICH ONE, (E) OR (I
)? ";
70 INPUT Q$
80 PRINT Q$
90 IF Q$="E" THEN LET M=1
100 IF Q$="I" THEN LET M=2
110 IF M<>1 AND M<>2 THEN GOTO
60
120 PRINT
130 PRINT "GIVE SIZE? ";
140 INPUT N
150 PRINT N
160 IF M=1 THEN LET M=N+1
170 IF M=2 THEN LET M=2*N
180 DIM X(N,M)
190 SCROLL
200 PRINT "ENTER COEFFICIENTS"
205 SCROLL
210 IF M=N+1 THEN PRINT "AND OU
TPUT ARRAY"
220 SCROLL
230 PRINT "ROW COLUMN"
240 SCROLL
250 FOR R=1 TO N
260 FOR C=1 TO N
270 PRINT " ";R;TAB 6;C,
280 INPUT X(R,C)
290 PRINT X(R,C)
295 SCROLL
300 NEXT C
310 IF M<>N+1 THEN GOTO 380
320 PRINT " ";R; " OUT",
350 INPUT X(R,M)
360 PRINT X(R,M)
370 GOTO 390
380 LET X(R,N+R)=1
390 SCROLL
400 NEXT R
410 SCROLL
420 PRINT "ANY CORRECTIONS (Y)?
"
430 INPUT Q$
440 IF Q$<>"Y" THEN GOTO 590
450 SCROLL
460 PRINT "WHICH ROW? ";
470 INPUT R
480 PRINT R
490 SCROLL

```

Continued Next Page

```

500 PRINT "WHICH COLUMN (OUT IS
N+1)? ";
510 INPUT C
520 PRINT C
530 SCROLL
540 PRINT "ENTER VALUE? ";
550 INPUT X(R,C)
560 PRINT X(R,C)
580 GOTO 410
590 CLS
600 REM FORWARD ELIMINATION
610 FOR C=1 TO N-1
620 LET A=0
630 IF X(C,C) <> 0 THEN GOTO 730
640 FOR R=C+1 TO N
650 IF X(R,C)=0 THEN GOTO 720
660 FOR I=1 TO M
670 LET T=-X(R,I)
680 LET X(R,I)=X(C,I)
690 LET X(C,I)=T
700 NEXT I
710 GOTO 730
720 NEXT R
730 LET A=X(C,C)
740 IF A=0 THEN GOTO 1240
750 FOR R=C+1 TO N
760 LET B=X(R,C)/A
770 IF B=0 THEN GOTO 810
780 FOR I=C TO M
790 LET X(R,I)=X(R,I)-B*X(C,I)
800 NEXT I
810 NEXT R
820 NEXT C
830 REM CHECK DETERMINATE
840 LET D=1
850 FOR I=1 TO N
860 LET D=D*X(I,I)
870 NEXT I
880 IF D=0 THEN GOTO 1240
890 REM BACKWARD ELIMINATION
900 FOR C=N TO 2 STEP -1
910 LET A=X(C,C)
920 FOR R=C-1 TO 1 STEP -1
930 LET B=X(R,C)/A
940 FOR I=C TO M
950 LET X(R,I)=X(R,I)-B*X(C,I)
960 NEXT I
970 NEXT R
980 NEXT C
990 REM PRINT RESULTS
1000 IF M=2*N THEN GOTO 1100
1010 PRINT "THE INPUT ARRAY IS:"
1020 PRINT
1030 FOR I=1 TO N
1040 PRINT "X-";I;" = ";X(I,M)/X
(I,I)
1050 NEXT I
1060 GOTO 1180
1100 PRINT "THE INVERTED MATRIX
IS:"
1110 PRINT
1120 PRINT "ROW COLUMN"
1130 FOR R=1 TO N
1135 PRINT
1140 FOR C=1 TO N
1150 PRINT " ";R;TAB 7;C;X(R,N+C
)/X(R,R)
1160 NEXT C
1170 NEXT R
1180 PRINT
1190 PRINT "DETERMINATE IS: ",D
1200 PRINT
1205 PRINT "RUN AGAIN (Y)?"
1210 INPUT Q$
1215 CLS
1220 IF Q$="Y" THEN RUN
1230 STOP
1240 PRINT "SINGULAR COEFFICIENT
MATRIX"
1250 PRINT "NO UNIQUE SOLUTION"
1260 GOTO 1200
SYNTACTIC SUM: 58033, 8K ROM

```

other sub-row, moving the entire row to the top of the sub-matrix if necessary, and performing the reduction. If the computer finds a row it cannot handle, the program recognizes the singularity of the coefficient matrix and stops. Completion of the forward reduction sequence produces an upper triangular coefficient matrix and its determinant is found as the product of the diagonal elements.

If the determinant is not zero, the system can be uniquely solved and backward elimination proceeds in a manner similar to the forward steps. Upon completion the answers are found by dividing each row element of the augmented matrix (which has been similarly operated upon) by the diagonal coefficient in the same row. The computer then prints the answers in logical sequence, but larger problems will require CONTINUES when the screen is full. Finally, the determinant is printed and you can re-run the program.

Since, as usual, one "keypunch" error can throw the whole program into a cocked hat, I include a couple of test examples after the program listing as well as a practical example of the use for simultaneous equations. Have fun!

Simultaneous Equations:

Suppose you have four programs, each containing a number of PAUSE statements (P), several loop (L) statements, several other operational program lines (O), and (N) characters to be printed. You time the completion of each program and solve for the time each element takes. You can then use the answers to predict how long similar programs will take to run.

Prog #	P	L	N	O	Time (sec)
1	15	26	18	31	6.2
2	28	17	15	23	5.4
3	19	42	7	36	2.9
4	11	21	12	19	4.2

The answers are: X-1 = .017664834 sec.

-2 = .011355478

-3 = .32093198

-4 = .0044190484

Matrix Inversion:

The following matrix is singular:

4	1	2	6
1	2	4	1
0	-1	-2	1
0	6	12	0

The inverse of the matrix:

1	0	1	is	0	0	1
3	1	0		0	1	-3
1	0	0		1	0	-1

50

ZX/TS

Home Budget

One of the first applications that come to mind when you get a personal computer is managing your home finances. However, when I got a copy of Sinclair's Home Budgeter, I was disappointed. It was fast, with good displays, but difficult to follow and modify. It seemed to do the wrong things. I wanted a program to keep track of my monthly expenditures in a variety of accounts as well as annual totals for expenditures by item and as a whole. This Budget version resulted.

Type in Home Budget as listed. Before using it the first time, save blank copies on tape by entering RUN 9900. These copies are self-running from then on.

To use Home Budget, just follow the prompts. The prompts should be adequate to avoid errors, but most input routines will check for an invalid input and reject it. If you do manage to stop the program for some reason, you can re-enter without losing your data by entering GOTO 1. Do not use RUN or you will lose all your data. To start over, enter GOTO 100. It's good to leave your program line cursor at line 1, so if the listing comes up on the screen you see a reminder not to RUN.

Home Budget handles up to 75 differently named accounts of up to 12 characters. When you first set up a budget, consider setting aside two or three extra accounts with no budget figures, in case you want to add an item or two to your budget later. You can call up accounts by any unique part of their name or their number.

After you set up your accounts the program gives you a menu (see Figure 1). This menu accesses all the program's functions.

My version of Home Budget offers some features you may like whether or not you need the whole program. The subroutine at lines 10-38 converts the floating point variable M to M\$, a dollar-and-cents figure as we like to view it. If M is negative, the routine at lines 20-26 changes the string to inverse characters. Printing in inverse makes the negative numbers stand out clearly in a list.

Lines like 1645 and 1660 provide another nice feature throughout the program, right-justifying the figures for easier reading and neater printout.

```

BUDGET NAME
1: SUMMARY OF ALL ACCOUNTS
2: DETAILS OF ONE ACCOUNT
3: ENTER NEW INFORMATION
4: CHANGE A BUDGETED AMOUNT
5: CHANGE/CORRECT THE NAME OF
   ACCOUNT
6: SAVE THE PROGRAM
7: EXIT THE PROGRAM
LAST UPDATE: NEW TAPE

```

Figure 1

I like to run Home Budget as listed, but you can get faster results by running your computer in FAST mode all the time. To run Home Budget in FAST, delete all SLOW and PAUSE statements. Change lines like 1695 to INPUT X\$.

Line 1 is the Bytes Remaining routine from SYNTAX newsletter, Oct. 82. I like to include it in most programs to see how much memory is left.

Lines 100-345 initialize and identify variables.

Ed. note: For ease in reading the listing, here are the texts of lines in inverse video:

Line	Text
2	space asterisk space asterisk space DO NOT RUN space asterisk space asterisk space asterisk 3 spaces IT WILL KILL ALL YOUR DATA 8 spaces USE 2 spaces "GOTO 1" 2 spaces TO RESTART 4 spaces COPYRIGHT (C) 2 spaces OCT. 7, 1982 2 spaces BY MICHAEL ROBERTS - DES MOINES, IA

4 space TO BEGIN A NEW FILE 3 spaces
 ENTER 2 spaces "RUN 100" 16 spaces
 110 ENTER THE NUMBER OF BUDGET
 ITEMS
 250 ENTER THE NAME OF THIS PROGRAM
 1610 ANNUAL BUDGET 4 spaces SPENT TO
 DATE 2 spaces
 1680 "Z"=COPY 2 spaces "C"=CONTINUE 2
 spaces "M"=MENU
 1730 MONTHLY BUDGET:
 1865 OVERSPENT:
 1880 "Z"=COPY 2 spaces "C"=CONTINUE 2
 spaces "M"=MENU
 2010 DETAILS OF AN ACCOUNT
 2040 BUDGET/MONTH:
 2350 9 spaces Z=COPY 3 spaces M=MENU 8
 spaces
 2510 6 spaces NEW INFORMATION ENTRY 5
 spaces
 2520 same as 2510
 2550 (ENTER "1"OR "2")
 2590 (USE THE 3 LETTER ABBREV.)
 2680 5 spaces ANOTHER ENTRY? (Y/N) 7
 spaces
 3010 CHANGE A BUDGET FIGURE FOR
 ACCT.
 3050 (ENTER "1"OR "2")
 3520 PRESENT NAME:
 3560 NEW NAME:
 4025 REMEMBER TO MAKE A "SPARE
 COPY" 1 space
 4550 NOT

SO

```

1 REM EORND FAST AT 5 T GOSU
B PI FAST AT TAN
2 REM * DO NOT RUN *
IT WILL KILL ALL YOUR DATA
USE GOTO 1 TO RESTART
COPYRIGHT 1981 OCT 1981
MICHAEL ROBERTS - DEE MOORE, JR
3 REM
4 REM TO BEGIN A NEW FILE
ENTER "RUN 100"
5 GOTO 1000
10 LET M=INT (M*100+.5)
12 LET M$=STR$ M
14 IF M=0 THEN LET M$="000"
16 LET M$=M$( TO LEN M$-2)+". "
+M$(LEN M$-1 TO )
18 IF VAL M$>=0 THEN GOTO 38
20 LET M$=M$(2 TO )
22 FOR Z=1 TO LEN M$
24 LET M$(Z)=CHR$ (CODE M$(Z)+
128)
26 NEXT Z
38 RETURN
100 LET O$="JANFEBMARAPR MAYJUNJ
ULAUGSEP OCTNOVDEC"
105 SLOW
110 PRINT "ENTER THE NUMBER OF
BUDGET ITEMS"
120 INPUT N1
130 PRINT N1
140 LET L$="

```

```

150 LET C$=""
160 LET S=1E3
170 LET G=10
180 LET L=5E3
190 LET F=4E4
200 LET E=21
210 LET D$="NEW TAPE"
250 PRINT "ENTER THE NAME OF TH
IS PROGRAM"
260 INPUT O$
270 IF LEN O$<=32 THEN GOTO 300
280 PRINT "THE NAME IS TOO L O
N G"
290 GOTO 260
300 PRINT O$
310 PRINT "NOW WE WILL ENTER TH
E NAMES AND AMOUNTS OF EACH BUDG
ET ITEM..."
315 REM **NAME OF ACCTS**
320 DIM N$(N1,15)
325 REM **DOLLAR FIGURES**
330 DIM D(N1,13)
335 REM **COLUMN TOTALS**
340 DIM T(14)
345 REM **ROW TOTALS**
350 FOR I=1 TO N1
360 PRINT AT E,31;AT E,0;"NAME
OF BUDGET ITEM NUMBER ";I;" "
370 IF I<10 THEN LET N$(I, TO 2
)="0"+STR$ I
380 IF I>=10 THEN LET N$(I, TO
2)=STR$ I
390 INPUT N$(I,4 TO )
400 SCROLL
410 PRINT AT E,31;AT E,5;N$(I)
420 SCROLL
430 PRINT AT E,31;AT 21,0;"ENTE
R MONTHLY BUDGET AMOUNT:"
440 INPUT D(I,1)
450 LET M=D(I,1)
460 GOSUB G
470 SCROLL
480 PRINT AT E,31;AT E,5;"$";M$
490 SCROLL
500 NEXT I
510 LET T=0
520 FOR I=1 TO N1
530 LET T=T+D(I,1)
540 NEXT I
550 CLS
560 PRINT "TOTAL BUDET REQUIRED
PER MONTH:"
570 LET M=T
575 GOSUB G
580 PRINT "$";M$
590 LET M=T*12
600 PRINT "TOTAL REQUIRED PER Y
EAR:"
605 GOSUB G
610 PRINT "$";M$
615 PAUSE 250
1000 REM **MENU**
1001 FAST
1002 CLS
1005 PRINT L$
1006 PRINT AT 0,0;L$;AT 0,0;"F";
AT 0,31;" ";
1007 FOR I=1 TO G
1008 PRINT " ";TAB 31;" ";TAB 3
1;TAB 31;" "
1009 NEXT I
1010 PRINT L$
1015 SLOW
1020 PRINT AT 1,1;O$
1030 PRINT TAB 1;" "
1040 PRINT AT 3,2;"1: SUMMARY OF
ALL ACCOUNTS"

```

Continued Next Page


```

1050 PRINT AT 5,2;"2: DETAILS OF
ONE ACCOUNT"
1060 PRINT AT 7,2;"3: ENTER NEW
INFORMATION"
1070 PRINT AT 9,2;"4: CHANGE A B
UDGETED AMOUNT"
1080 PRINT AT 11,2;"5: CHANGE/CO
RRECT THE NAME OF";TAB 5;"ACCOUN
T"
1085 PRINT AT 14,1;"*****"
1090 PRINT AT 15,2;"6: SAVE THE
PROGRAM"
1100 PRINT AT 17,2;"7: EXIT THE
PROGRAM"
1101 PRINT AT 19,1;"*****"
1105 PRINT TAB 2;"LAST UPDATE: "
;D$
1110 PAUSE F
1120 LET X$=INKEY$
1130 IF X$="" OR CODE X$<26 OR C
ODE X$>35 THEN GOTO 1110
1135 CLS
1140 GOTO S+500*VAL X$
1500 REM **SUMMARY**
1510 FAST
1520 DIM S(N1)
1525 DIM T(14)
1530 FOR I=1 TO N1
1535 FOR J=2 TO 13
1540 LET S(I)=S(I)+D(I,J)
1545 NEXT J
1550 NEXT I
1560 FOR I=1 TO 13
1565 FOR J=1 TO N1
1570 LET T(I)=T(I)+D(J,I)
1575 NEXT J
1580 IF I>1 THEN LET T(14)=T(14)
+T(I)
1585 NEXT I
1590 SLOW
1600 FOR I=1 TO N1 STEP G
1605 FOR J=0 TO 9
1610 IF J=0 THEN PRINT AT 0,0;"B
UDGETED BUDGET SPENT TO DATE"
1615 IF J+I>N1 THEN GOTO 1680
1620 PRINT N$(J+I)
1630 PRINT TAB 6;"$";
1635 LET M=12*D(I+J,1)
1640 GOSUB G
1645 PRINT TAB (15-LEN M$);M$,"
$";
1650 LET M=S(I+J)
1655 GOSUB G
1660 PRINT TAB (28-LEN M$);M$
1670 NEXT J
1680 PRINT AT E,0;"Z=COPY"
1685 PRINT AT E,0;"*****"
1690 PAUSE F
1695 LET X$=INKEY$
1700 IF CODE X$=118 OR X$="M" TH
EN GOTO S
1705 IF X$="Z" THEN PRINT AT 21,
0;"
1710 IF X$="Z" THEN COPY
1715 IF X$="Z" THEN GOTO 1680
1720 CLS
1725 NEXT I
1730 PRINT "MONTHLY BUDGET $";
1735 LET M=T(1)
1740 GOSUB G
1745 PRINT M$
1750 PRINT
1755 PRINT "SPENT IN";TAB 12;"AM
OUNT";TAB 20;"*****"=BALANCE";TAB
0;"MONTH OF";TAB 12;"SPENT";TAB
20;"*****"=DEFICIT"

```

```

1757 PRINT "
"
1760 FOR I=2 TO 13
1765 LET M=T(I)
1770 GOSUB G
1775 PRINT TAB 2;0$((I-2)*3+1 TO
(I-2)*3+3);TAB 10;"$";TAB (19-L
EN M$);M$;
1780 LET M=T(1)-VAL M$
1785 GOSUB G
1786 IF M>=0 THEN PRINT TAB 20;"
+";
1787 IF M<0 THEN PRINT TAB 20;"-
";
1790 PRINT TAB (31-LEN M$);M$
1795 NEXT I
1799 LET T=0
1800 FOR I=2 TO 13
1805 LET T=T+T(I)
1810 NEXT I
1820 PRINT ", "TOTAL ANNUAL BUDGE
T: $";
1825 LET M=12*T(1)
1830 GOSUB G
1835 PRINT TAB (31-LEN M$);M$
1840 LET M=T
1845 GOSUB G
1850 PRINT "TOTAL SPENT TO DATE:
$";TAB (31-LEN M$);M$
1855 LET M=12*T(1)-T
1860 IF M>=0 THEN PRINT "BALANCE
LEFT: "
1865 IF M<0 THEN PRINT "OVERSPEN
"
1870 GOSUB G
1875 PRINT TAB 21;"$";TAB (31-LE
N M$);M$
1880 PRINT AT 21,0;"Z=COPY"
1885 PRINT AT 21,0;"*****"
1890 PAUSE F
1895 LET X$=INKEY$
1900 IF CODE X$=118 OR X$="M" TH
EN GOTO 1000
1905 IF X$="Z" THEN PRINT AT E,0
;"
1910 IF X$="Z" THEN COPY
1915 IF X$="Z" THEN GOTO 1880
1920 GOTO S
2000 REM **ANALYSIS**
2010 PRINT "DETAILS OF AN ACCOUN
T"
2020 GOSUB L
2030 PRINT Y$
2040 PRINT "BUDGET/MONTH: $";
2045 LET M=D(Y,1)
2050 GOSUB G
2055 PRINT M$
2060 PRINT
2065 PRINT "MON AMT SPENT BALAN
CE ACCUM BAL"
2070 PRINT L$
2075 LET T=D(Y,1)
2080 LET J=0
2085 FOR I=2 TO 13
2090 PRINT 0$((I-2)*3+1 TO (I-2)
*3+3);
2100 LET M=D(Y,I)
2105 LET J=J+M
2110 GOSUB G
2120 PRINT TAB 12-LEN M$;M$;
2130 LET M=T-D(Y,I)
2140 GOSUB G
2150 IF M>=0 THEN PRINT TAB 13;"
+";
2160 IF M<0 THEN PRINT TAB 13;"-
";
2180 PRINT TAB 22-LEN M$;M$;
2185 LET M=T*(I-1)-J

```

Continued Next Page

```

2190 IF M<0 THEN PRINT TAB 23;"
2195 IF M>=0 THEN PRINT TAB 23;"
2200 GOSUB G
2210 PRINT TAB 32-LEN M$;M$
2215 NEXT I
2219 PRINT
2220 PRINT "ANNUAL BUDGET:","$";
2225 LET M=T*12
2230 GOSUB G
2240 PRINT TAB 25-LEN M$;M$
2250 PRINT "SPENT TO DATE:","$";
2260 LET M=J
2270 GOSUB G
2280 PRINT TAB 25-LEN M$;M$
2290 PRINT "BALANCE:";
2300 LET M=T*12-J
2310 IF M>=0 THEN PRINT "+";
2320 IF M<0 THEN PRINT "-";
2330 GOSUB G
2340 PRINT TAB 25-LEN M$;M$
2350 PRINT AT 21,0;"
2360 PAUSE F
2365 LET Y$=INKEY$
2375 IF Y$="Z" THEN COPY
2380 IF Y$="Z" THEN GOTO 2350
2390 CLS
2400 GOTO S
2500 REM **NEW INFO**
2505 CLS
2510 PRINT "
2515 GOSUB L
2520 PRINT "
2525 PRINT Y$
2530 PRINT "WILL YOU ENTER . .
2540 PRINT "1: MONEY SPENT FROM THIS ACCOUNT";TAB 14;" OR "2
: MONEY TO BE CREDITED TO THIS ACCOUNT"
2550 PRINT TAB 6;"ENTER
2555 PAUSE F
2560 LET X$=INKEY$
2570 IF X$<>"1" AND X$<>"2" THEN
GOTO 2555
2575 LET X=VAL X$
2580 PRINT "WHAT MONTH IS THE ENTRY FOR?"
2590 PRINT "USE THE 3 LETTER
2600 INPUT Y$
2605 FOR I=0 TO 11
2610 IF Y$=0$((I*3)+1 TO (I*3)+3) THEN GOTO 2625
2615 NEXT I
2620 PRINT "WHAT?"
2621 GOTO 2590
2625 LET I=I+2
2630 PRINT "ENTER THE AMOUNT:"
2640 INPUT J
2650 IF J<0 THEN GOTO 2590
2660 IF X=2 THEN LET J=-J
2670 LET D(Y,I)=D(Y,I)+J
2680 PRINT AT 21,0;"
2690 PAUSE F
2700 IF INKEY$="Y" THEN GOTO 250
2710 GOTO S
2990 STOP
3000 REM **CHANGE BUDGET FIGURE*
3005 GOSUB L
3010 PRINT "CHANGE A BUDGET FIGURE FOR
3020 PRINT Y$
3030 PRINT "WILL YOU ENTER . .

```

```

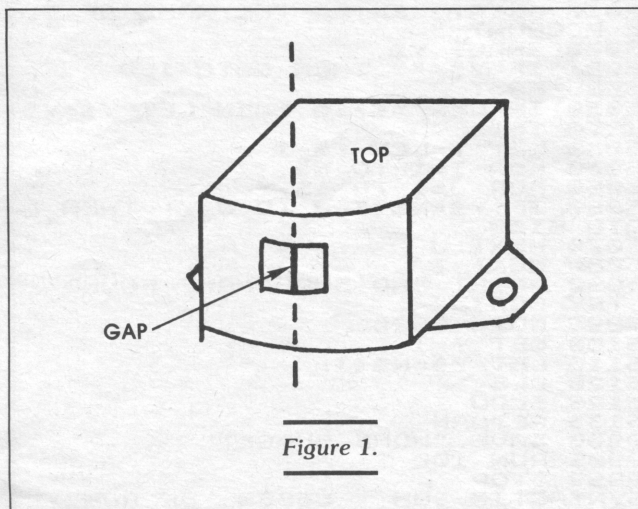
3035 PRINT "1: A NEW MONTHLY FIGURE";TAB 14;" OR "
3040 PRINT "2: A CHANGE IN THE ANNUAL TOTAL"
3050 PRINT TAB 6;"ENTER
3060 PAUSE F
3070 LET X$=INKEY$
3075 IF X$<>"1" AND X$<>"2" THEN
GOTO 3070
3080 LET X=VAL X$
3090 PRINT "ENTER THE AMOUNT:"
3100 INPUT J
3110 IF X=1 THEN GOTO 3200
3120 LET J=J/12
3130 LET J=INT (J*100)/100
3140 LET D(Y,1)=D(Y,1)+J
3150 GOTO S
3200 LET D(Y,1)=J
3205 GOTO S
3500 REM **CHANGE NAME**
3505 PRINT "TO CHANGE AN ACCOUNT NAME, FIRST"
3510 GOSUB L
3520 PRINT "PRESENT NAME ";Y$
3530 PRINT "ENTER THE NEW NAME
3540 INPUT Y$
3550 LET N$(Y,4 TO )=Y$
3560 PRINT "NEW NAME ";N$(Y)
3570 PAUSE F
3580 GOTO S
4000 REM **SAVE**
4010 PRINT "ENTER TODAY'S DATE:"
4015 INPUT D$
4020 PRINT AT 5,0;"PREPARE TAPE RECORDER
4025 PRINT AT 8,0;"REMEMBER TO
4030 PRINT AT 21,0;"PRESS ENTER TO START"
4035 PAUSE F
4040 SAVE D$
4050 GOTO S
4500 REM **EXIT**
4510 PRINT "ARE YOU SURE YOU WANT TO QUIT THE PROGRAM WITHOUT SAVING ANY NEW INFORMATION?"
4520 PAUSE F
4530 IF INKEY$<>"Y" THEN GOTO 10
4550 PRINT AT 10,0;"IF YOU CHANGE YOUR MIND, ENTER "GOTO 1000"
DO NOT PRESS RUN."
4560 GOTO 9999
5000 REM **SEARCH**
5010 PRINT "ENTER THE NAME OF THE ACCOUNT:"
5020 INPUT Y$
5021 IF Y$="" THEN GOTO 1E3
5025 FAST
5030 IF LEN Y$>15 THEN LET Y$=Y$ ( TO 15)
5035 LET Y=LEN Y$-1
5040 FOR I=1 TO N1
5050 FOR J=1 TO 15-Y
5060 IF Y$=N$(I,J TO J+Y) THEN GOTO 5100
5070 NEXT J
5080 NEXT I
5090 PRINT "NO SUCH NAME FOUND - TRY AGAIN"
5095 GOTO 5020
5100 LET Y=I
5110 LET Y$=N$(I)
5120 CLS
5125 SLOW
5130 RETURN
9900 SAVE "HOME BUDGET"
9905 RUN 100
9999 STOP
SYNTACTIC SUM: 55389, 8K ROM

```


Getting Your Head Straight

If you're not a tape recorder buff, you may not know the term azimuth as it applies to recorders. If you do, you may have some inkling as to what follows. In either case, you may learn something crucial to computing and storing programs on tape.

In simple terms, "azimuth" is a line of direction. "Azimuth alignment" related to tape recorders refers to the line of direction of the gap in the record head pole pieces (or iron). Sometimes you can't see this tiny gap with the naked eye, but it is always there. (In some modern recorders the gap is filled with ceramic or other material to prevent tape oxides from getting into the gap, causing a magnetic short and subsequent loss of volume. This problem can cause recording malfunction. For the moment, let's stick with azimuth alignment.) Figure 1 illustrates a record/play head, locating the gap and azimuth.



Note that the gap runs vertically (top to bottom) on the head at right angles to the direction of tape travel. This is the crux of the matter. Just how precisely you maintain this 90-degree relationship determines the high frequency

response of the recordings. With voice recording, high frequency response is not critical. With music, it depends on how important you think it is to hear the high pitch tones in your music. With computer data tapes, however, it is crucial that high frequencies do not get lost. Computer data on tape is all relatively high frequency. Your tape equipment must respond properly to the high frequency signals from the computer circuits or data will be lost and you will have problems.

So, right about now, some of you lucky ones are saying "Well, that's all very interesting but my tapes SAVE and LOAD fine. So why should this concern me?" Well, here are some reasons why.

If you never intend to exchange tapes with other people, then azimuth alignment is not such a concern. As long as tapes are recorded and played on the same recorder, even with poor azimuth alignment, the head alignment appears good to the tape. But, put that same tape on a recorder with different azimuth angle and the high frequencies will be reduced—maybe just slightly or completely lost. At some point, you can't load from the tape anymore. Your ability to buy or trade taped programs is seriously impaired unless everybody involved has correct azimuth alignment on their recorders.

Also, if you put off aligning your recorder head you will create a giant re-recording problem for yourself. If you change your azimuth down the line somewhere, you won't be able to load previously recorded tapes. Obviously the best time to check and realign is as soon as possible so you don't build up an extensive tape library prior to setting your heads straight.

What it boils down to is this—any tape not recorded or played back on machines with correct (90-degree) azimuth alignment will probably work poorly, if at all.

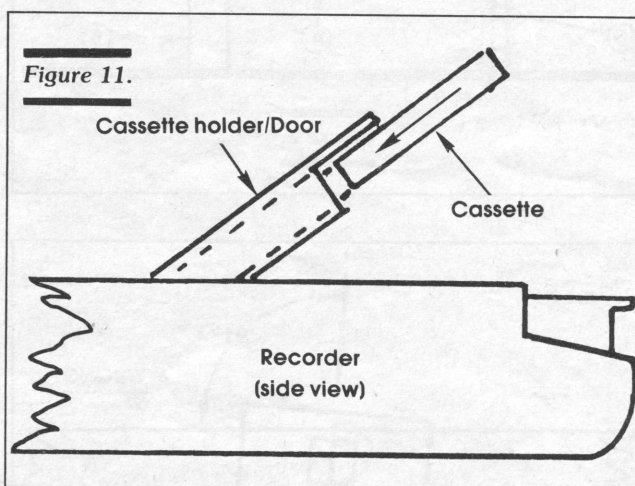
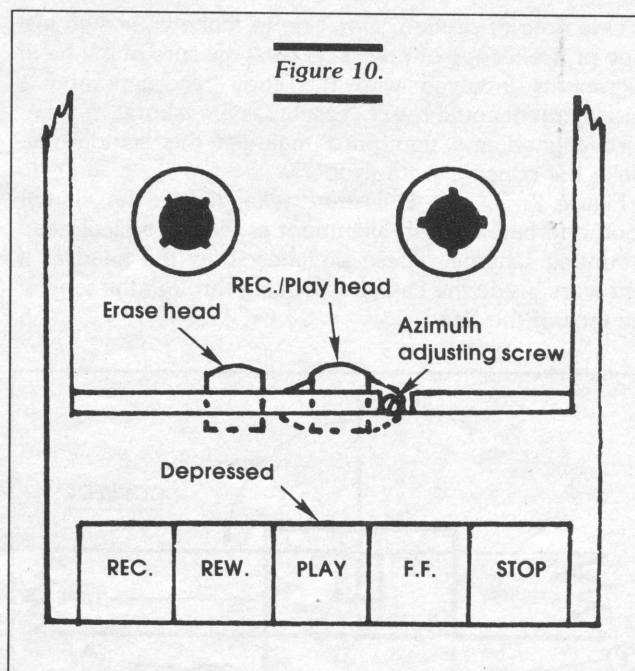
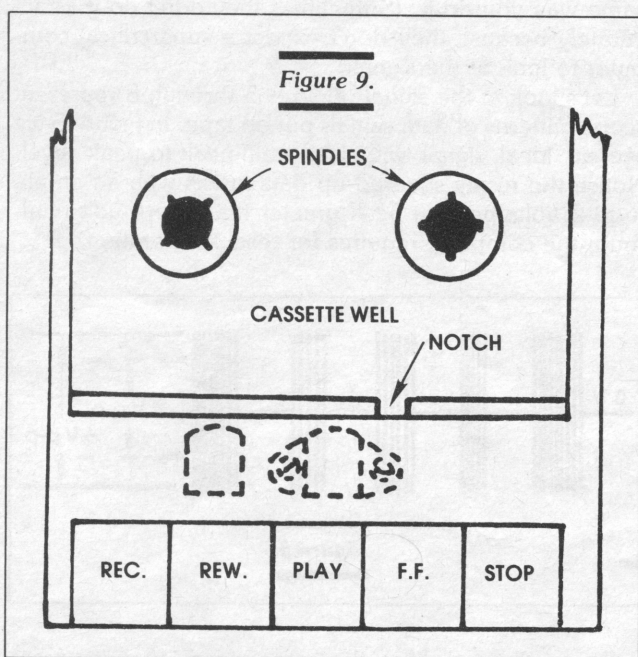
For best results, before you proceed with alignment, clean and demagnetize the heads.

Now, the question is, how to align your azimuth? Several companies market test tapes made specifically for this function. The most likely place to get one is a store catering to audiophiles. A test cassette typically provides several recorded tests, including timing, wow and flutter

as well as azimuth alignment. The alignment portion consists of several tones preceded by voice announcements of their frequencies. They begin with relatively low tones and increase with each test. For example: test one might be a 500-cycle tone; test two, 1000 cycles; test three, 3000 cycles, and so on up to 10,000 or 15,000 cycles. Play a test track on your recorder and listen to the tones after each announcement. If you can't hear the tone after an announcement you have passed the limit of your recorder's high frequency response. If this occurs at a relatively low frequency, say 2000 cycles, you have a problem. If you can hear 5000 cycles or above, you are in pretty good shape.

Due to wide variations in recorder design, it is impossible to cover all possible variations here, so a general illustration will show you where to find pertinent items on recorders. You may have to adjust your thinking to relate the illustrations to your specific recorder's physical layout. In this regard, the basic problem is gaining access to the adjusting screw (E).

Figures 9 and 10 are top views of a typical recorder without a cassette inserted. Figure 9 shows how the heads retract (dashed lines) under the housing top when



no keys are depressed. It also shows the access notch (or hole) to the head adjusting screw. Figure 10 shows how the heads move out into the cassette well when you press the play key. When heads are in this position the adjusting screw is accessible through the notch (or hole) in the housing. On some recorders without the notch you reach the screw by inserting a tiny screw driver on a slant past the edge of the tape well.

If on your recorder, you insert the cassette in a trap-door and then close the door (Figure 11), there almost has to be an access hole because the entire top surface closes to play the tape. Sometimes a metal foil label covers the hole. Peel up the label to gain access.

If you find no access to the screw (E), you can do one of two things: remove the recorder housing, make the adjustment, then reassemble (*NOT* recommended for novices); or make a hole (this is easier, but be careful).

To make a hole, locate the screw position exactly with the heads in play position (Figure 10) and mark the location. Retract the heads (Figure 9) and make a hole in the housing at the marked location. Use great care not to damage the head or mechanism in the process—do not let chips fall into the recorder, make them fall away from it.

To adjust the azimuth: play the test track with the highest tone you can hear (test tapes come with user instructions). As it plays, adjust the tilt of your recorder's head to the point that gives the loudest volume. Then, play the track you couldn't hear before. If you can hear it now adjust the head tilt during this track for maximum volume. When you achieve the *loudest* volume on the *highest* tone track you can hear, you have correct azimuth alignment.

One note of caution. Don't try to make your own test tape or use a copy of one. You can't be sure of the head alignments involved with the copy recorders and a misalignment could result. Test tapes are laboratory standard aligned and you must maintain this standard to make the concept work properly.

Figure 2, 3, and 4 illustrate (exaggerated for clarity) good and bad azimuth alignment as well as typical head mounting scheme. These sketches show the head as if you were inside the cassette, looking through the tape at the face of the head.

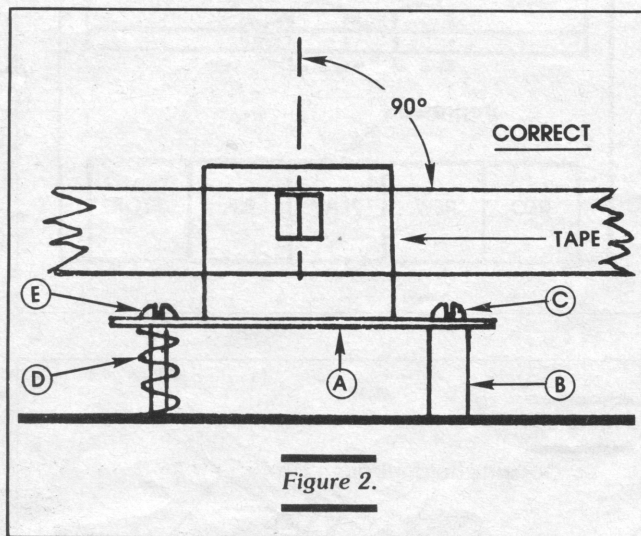


Figure 2.

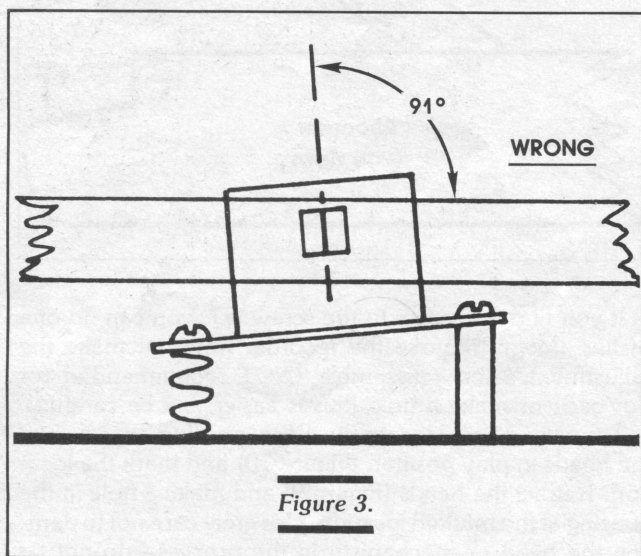


Figure 3.

Head mounting flange A fastens to spacer post B with screw C. Spring D replaces spacer post B on the opposite side of the head. Adjusting screw E up or down against the spring changes the tilt of the head assembly. Thus, the azimuth angle changes relative to the tape edge.

Adjusting screw E would normally not require more than a half turn in either direction to correct the azimuth alignment. Usually a very slight amount of turning will do the trick—about one-eighth of a turn is ample.

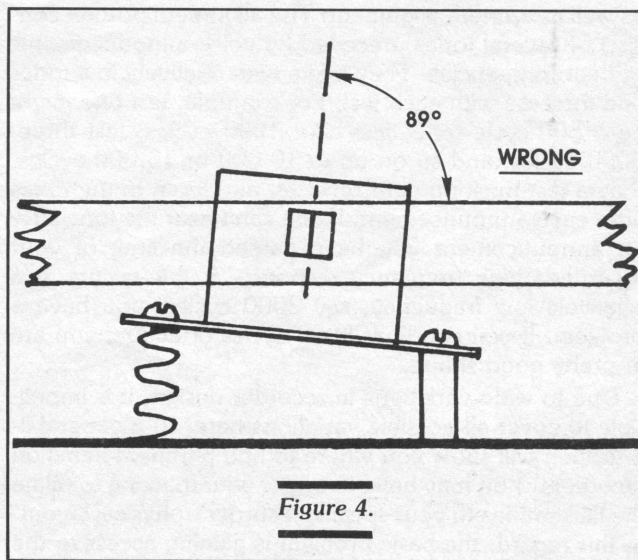


Figure 4.

In some cases spacer B and spring D may be reversed, or the whole setup may differ, but the principle remains the same. The manufacturer also sets the azimuth the same way you do it. (Sometimes they don't do it as accurately because they don't expect a supercritical computer to look at the signal.)

Let's look at the signal. Figures 5 through 8 represent scope patterns of data signals put on tape. In Figure 5 we see an ideal signal with a 6 volt peak-to-peak level. Notice the nicely squared-up data bursts with an amplitude 2 volts peak-to-peak greater than the 4-volt minimum the computer requires for reliable operation.

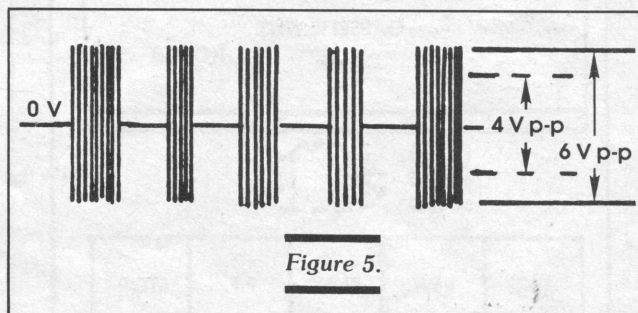


Figure 5.

Figure 6 exemplifies the same signal after it passes through a recorder with a slightly different azimuth alignment from the one used for Figure 5. Note the tapered beginnings of the top and bottom edges of the signals.

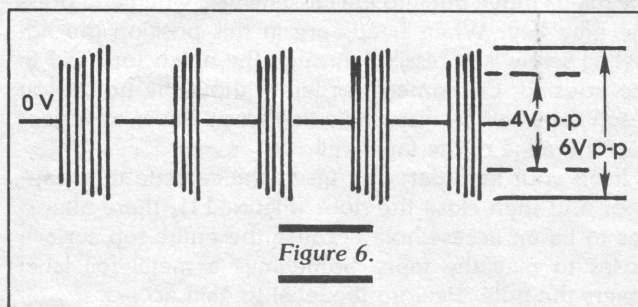


Figure 6.

This is known as high frequency roll-off and is caused by the inability of the recorder/amplifier and/or record/play head to reproduce the abrupt signal level change of a square wave signal. Notice the tapered part of the signal still exceeds 4 volts, so this amount of roll-off would not cause data loss.

Move on to Figure 7. This illustrates a more severe case of azimuth misalignment. The roll-off now reduces portions of the signals below the 4-volt level. Now you lose data and experience LOADING problems.

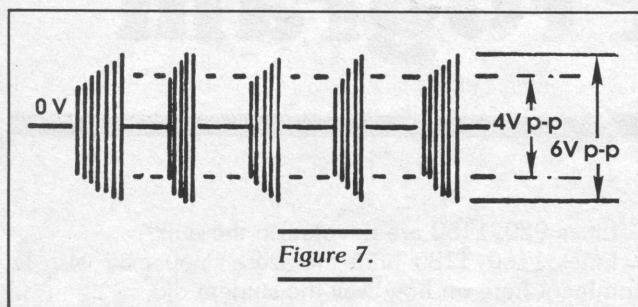
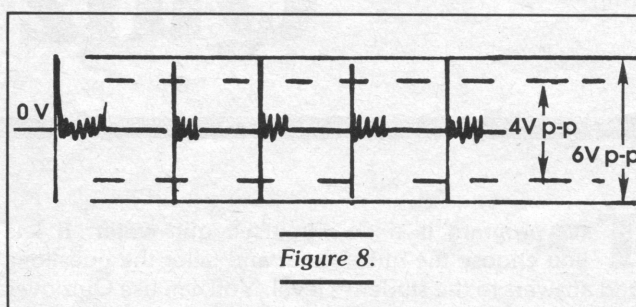


Figure 8 illustrates severe misalignment resulting in almost total loss of data bursts and leaving just some useless spiking signals which the computer ignores.

Other problems also cause poor performance of cassette machine and tapes, such as tape oxide buildup on head gap, and poor oxide coating on tapes and noisy recorders. These problems can be overcome rather easily. However, if we want to build a ZX80/81 and Timex users program exchange capability we need to "get our heads straight" to make it work out.



"IF I HAD TO CHOOSE JUST ONE PROGRAM TO IMPRESS AN AUDIENCE, THEN '3D MONSTER MAZE' WOULD BE THE ONE"*

Review from
ZX COMPUTING

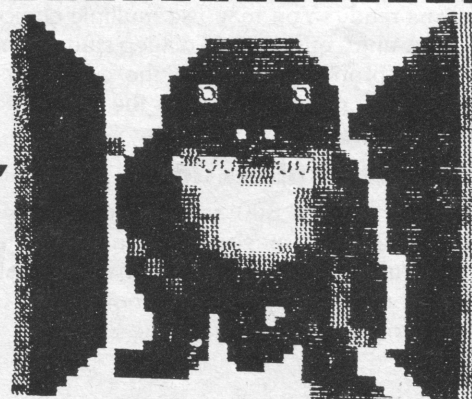
	Price	Quantity	Total
13941 Gamestape 1: 11 Programs—1K	14.95		\$
13942 Gamestape 2: 3 Games—16K	14.95		
13943 Gamestape 3: Catacombs Adventure—16K	14.95		
13944 Gamestape 4: 3D Monster Maze—16K	14.95		
13945 Gamestape 5: 3D Orbiter—16K	14.95		
PLUS BOOKS:			
26025 NOT ONLY 30 PROGRAMS FOR THE ZX81	14.95		
25957 MACHINE LANGUAGE PROGRAMMING MADE SIMPLE	19.95		
25913 UNDERSTANDING YOUR ZX81 ROM	19.95		
26063 ZX81 ROM DISASSEMBLY—PART A	14.95		
26103 ZX81 ROM DISASSEMBLY—PART B	14.95		
25895 COMPLETE BASIC COURSE FOR TS1000/ZX81	34.50		
PLUS TAPES:			
26446 SPACE TREK —16K	14.95		
26359 SUPER INVASION —1K	14.95		
26318 WALL BUSTERS —1K	14.95		
26472 10 EXCITING PROGRAMS —1K	14.95		
26284 REVERSI —1K	14.95		
26406 TOOLBOX —1K	14.95		
26490 BASIC COURSE 2 cassette pack	7.50		
Total:			\$
Residents of CA, MD, TN, please add sales taxes:			
S&H:			2.00
Total:			\$

Enclosed is my check or money order for \$ _____.

Please charge my Visa _____ or MasterCard _____.

Card # _____ expiration date _____.

**Visa and M/C orders
can be phoned in:
615/361-3738**



3D MONSTER MAZE
Gamestape 4 Actual screen TS1000 / ZX81

Orders to:
**MELBOURNE HOUSE
SOFTWARE, INC.**
Dept. CS
347 Reedwood Drive
Nashville, TN 37217



Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Dealer orders and queries:

800/251-5900
(ask for a Melbourne House operator)

STX

Make-It-Yourself Quiz Program

This program is a do-it-yourself quiz-writer. It lets you choose the quiz format and tailor the questions and answers to the student's level. You can use Quiz over and over for different tests.

Enter Quiz as listed. Before running the program, SAVE it on tape. This tape is now your master copy. Whenever you want to make new quizzes, just load your master tape.

To use Quiz, just press RUN and ENTER (or NEWLINE on older machines). Have your questions and answers ready. You may use multiple choice, true-false, or direct-entry questions. To keep students from cheating on math problems by using the computer to calculate answers, the program requires the student to solve math problems before entry.

Program Explanation

Lines 10 through 300 set up the variables. Line 230 calculates how many characters the machine can devote to your questions. The program will use almost all the available memory. If you have more than 16K RAM and don't want to use it, change line 230 to:

```
230 LET E=9600-(D*A+1)*5)
```

This change will limit the program to about 16K. Line 250 displays the number of questions you requested, then gives you the total character count for your questions. It also tells you the average number of characters you can use for each question. This number includes the question, answer, choices and any explanation you wish. Use this number as a guide to plan your entries and avoid running out of memory.

Lines 310-760 let you input your questions, answers, choices and any statement you want to make to the student about the question he/she just answered incorrectly.

Lines 770-820 display the quiz title.

Lines 830-910 start the quiz for the student and provide user-friendly greetings.

Lines 920-1150 are devoted to the quiz.

Lines 1160-1180 print the score. You may wish to comment here on how well the student did.

Lines 1190-1240 give you the option of saving the quiz program on tape. It will not retain the student's name or the score. If you have already saved Quiz with this question/answer set, press ENTER to go back to the beginning of the quiz.

Quiz is a loop program. To exit you have to BREAK either at one of the PAUSEs or during a false SAVE routine (you tell it to SAVE, but recorder is not turned on). If you accidentally BREAK and wish to continue, just enter CONT. If that fails, enter GOTO 1160. You will be able to restart your quiz from the beginning without losing your questions. If you should BREAK during your question and answer input, try CONT. If that doesn't work, you'll have to start over by entering RUN.

(Ed. Note—When you enter Quiz, follow the spacing in the listing exactly. It is important for a nice display.

Lines with inverse video and graphics are:

Line	Graphics	Inverse Video
30	32 graphic shift 7	ENTER ANSWER with 1 space on either side
900		ZX-81 QUIZ with 1 space on either side
1020		VERY GOOD THAT IS CORRECT with 1 space on either side
1060		I AM SORRY THAT IS NOT CORRECT with 1 space on either side
1130		ENTER with 1 space on either side
1170		CORRECT with 1 space on either side

After entering your questions and answers, you can BREAK to stop program execution. Start your tape recorder, then enter GOTO 1230. The program will save to tape under the name "QUIZ." When you next load the tape, the program will start automatically with the quiz at line 830. If you BREAK now and look at the program listing, you'll see the Z in QUIZ (line 1230) is in inverse video. This is normal in self-starting programs.)

List of Variables Used

DIM A\$(E)	Stores all questions, answers, choices and statements.
DIM A (A,D)	Used to slice DIM A\$(E).
B\$	See program line 30.
C\$	Sets up the type of test.
D\$	Sets up the wrong answer explanation.
E\$	Input information.
F\$	Input information.
H\$	Explanation of wrong answer.
I\$	Quiz title.
J\$	Student name.
K\$	Used to compare correct answer to student's answer.
A	Number of questions (length of array A).
B	Number of choices in multiple choice section.
C	FOR-NEXT routine counter.
D	Depth of array A.
E	Length of array A\$.
F	FOR-NEXT routine counter.
H	Used to calculate the length of strings.
I	FOR-NEXT routine counter.
S	Score.

SO

An enhanced version of this program is available on C-20 tape from the author for \$10. D Lipinski Software, 2737 Susquehanna Road, Roslyn, PA 19001.

```

5 REM "QUIZ"
10 LET S=0
20 FAST
30 LET B$="ENTER ANSWER"
40 CLS
50 PRINT "WHAT TYPE OF QUIZ DO YOU WISH TO HAVE?","1 = MULTIPLE CHOICE","2 = TRUE OR FALSE","3 = DIRECT ANSWER",B$
60 INPUT C$
70 IF CODE C$<=26 OR CODE C$>=32 THEN GOTO 40
80 CLS
90 PRINT "HOW MANY QUESTIONS DO YOU WISH TO ASK?",B$
100 INPUT A
110 CLS
120 PRINT "DO YOU WISH TO EXPLAIN IN THE ANSWER IF IT IS WRONG?"
130 INPUT D$
140 IF CODE D$<=27 OR CODE D$>=30 THEN GOTO 110
150 IF D$="0" THEN LET H$=" "
170 CLS
180 LET D=4
190 IF C$="2" OR C$="3" THEN GO TO 230

```

```

200 PRINT "WHAT IS THE MAXIMUM NUMBER OF","CHOICES PER QUESTION?",B$
210 INPUT B
215 IF B>=10 THEN GOTO 170
220 LET D=B+4
230 LET E=(PEEK 16386-PEEK 16412+256*(PEEK 16387-PEEK 16413)-50)-1300-(D*(A+1)*5)
240 CLS
250 PRINT "YOU WILL HAVE ",A," QUESTIONS","WITH A TOTAL CHARACTER INPUT OF",E,TAB 0;"AN AVERAGE OF ",E/A,"CHARACTERS PER QUESTION";TAB 0;"IF THIS IS OKAY ENTER 0","IF NOT ENTER 1",B$
260 INPUT E$
270 IF CODE E$<=27 OR CODE E$>=29 THEN GOTO 40
280 DIM A(A+1,D)
290 DIM A$(E)
300 LET A(1,1)=1
320 FOR F=1 TO A
330 CLS
340 PRINT "WHAT IS QUESTION NUMBER ",F,"?",B$
350 INPUT E$
355 CLS
360 PRINT E$,TAB 0;"INPUT 0 IF THIS IS CORRECT","INPUT 1 IF NOT CORRECT",B$
370 INPUT F$
380 IF CODE F$<=27 OR CODE F$>=29 THEN GOTO 330
390 LET H=LEN E$
400 LET A(F,2)=A(F,1)+H
410 LET A$(A(F,1) TO A(F,1)+H)=E$
420 CLS
430 PRINT "WHAT IS THE ANSWER?"
440 IF C$="2" THEN PRINT "1 = TRUE","0 = FALSE"
450 PRINT B$
460 INPUT E$
465 CLS
470 PRINT E$,TAB 0;"INPUT 0 IF THIS IS CORRECT","INPUT 1 IF NOT CORRECT",B$
475 IF C$="2" AND E$="1" THEN PRINT AT 0,0;"1 = TRUE"
477 IF C$="2" AND E$="0" THEN PRINT AT 0,0;"0 = FALSE"
480 INPUT F$
490 IF CODE F$<=27 OR CODE F$>=29 THEN GOTO 420
495 IF C$="1" THEN LET E$=" "+E$
500 LET H=LEN E$
510 LET A(F,3)=A(F,2)+H
520 LET A$(A(F,2) TO A(F,2)+H)=E$
530 IF CODE D$<=26 OR CODE D$>=30 THEN GOTO 600
540 CLS
550 PRINT "WHAT IS THE EXPLANATION","NO EXPLANATION INPUT A SPACE",B$
560 INPUT H$
565 CLS
570 PRINT H$,TAB 0;"INPUT 0 IF THIS IS CORRECT","INPUT 1 IF NOT CORRECT",B$
580 INPUT F$
590 IF CODE F$<=27 OR CODE F$>=29 THEN GOTO 540
600 LET H=LEN H$
610 LET A(F,4)=A(F,3)+H
620 LET A$(A(F,3) TO A(F,3)+H)=H$
630 IF CODE C$<=26 OR CODE C$>=30 THEN GOTO 750
640 FOR C=1 TO B
650 CLS
660 PRINT "WHAT IS THE NUMBER ",C," CHOICE?","FOR NO CHOICE INPUT A SPACE",B$
670 INPUT E$

```

Continued Next Page


```

675 CLS
680 PRINT C;" = ";E$;TAB 0;"INPUT
UT 0 IF THIS IS CORRECT","INPUT
1 IF NOT CORRECT",B$
690 INPUT F$
700 IF CODE F$<=27 OR CODE F$>=
29 THEN GOTO 650
705 LET E$=CHR$(C+28)+" = "+E$
710 LET H=LEN E$
720 LET A(F,4+C)=A(F,3+C)+H
730 LET A$(A(F,3+C) TO A(F,3+C)
+H)=E$
735 IF A$(A(F,2)+1 TO A(F,3)-1)
=E$(2 TO ) THEN LET A$(A(F,2) TO
A(F,2))=CHR$(C+28)
740 NEXT C
750 LET A(F+1,1)=A(F,D)
760 NEXT F
770 CLS
780 PRINT "WHAT IS THE TITLE OF
THE QUIZ?",B$
790 INPUT I$
795 CLS
800 PRINT I$;TAB 0;"INPUT 0 IF
THIS IS CORRECT","INPUT 1 IF NOT
CORRECT",B$
810 INPUT F$
820 IF CODE F$<=27 OR CODE F$>=
29 THEN GOTO 770
830 CLS
840 PRINT "WHAT IS YOUR NAME PL
EASE?",B$
850 INPUT J$
860 CLS
870 PRINT "THANK YOU ";J$;TAB 0
;"GET READY FOR YOUR QUIZ"
880 PAUSE 400
890 CLS
900 PRINT TAB 10;"EX-81 QUIZ"
;TAB 0;I$;TAB 0;"TAKEN BY ";J$
910 PAUSE 400
920 FOR C=1 TO A
930 CLS
940 LET E$=A$(A(C,1) TO A(C,2)-
1)
945 LET K$=A$(A(C,2) TO A(C,3)-
1)
950 PRINT "QUESTION NUMBER ";C;
TAB 0;E$;TAB 0;B$( TO 32)
955 IF C$="1" THEN LET K$=A$(A(
C,2) TO A(C,2))
960 IF C$="1" THEN FOR I=4 TO D
-1
970 IF C$="1" THEN PRINT A$(A(C
,I) TO A(C,I+1)-1)
980 IF C$="1" THEN NEXT I
990 IF C$="2" THEN PRINT "1 = T
RUE", "0 = FALSE"
1000 PRINT B$
1010 INPUT F$
1020 IF K$=F$ THEN PRINT "VERY
GOOD THAT IS CORRECT"
1030 IF K$=F$ THEN PAUSE 400
1040 IF K$=F$ THEN LET S=S+1
1050 IF K$=F$ THEN GOTO 1150
1060 PRINT "I AM SORRY THAT IS
NOT CORRECT"
1070 PAUSE 400
1080 CLS
1090 PRINT "THE CORRECT ANSWER I
S";TAB 0;A$(A(C,2) TO A(C,3)-1)
1100 IF C$="2" AND K$="1" THEN P
RINT AT 1,0;"1 = TRUE"
1110 IF C$="2" AND K$="0" THEN P
RINT AT 1,0;"0 = FALSE"
1120 PRINT A$(A(C,3) TO A(C,4)-1
)
1130 PRINT "PRESS ENTER TO CON
TINUE",B$
1140 INPUT E$
1150 NEXT C
1160 CLS
1170 PRINT "QUIZ IS COMPLETE. TH
ANK YOU",J$;TAB 0;"YOUR SCORE I
S ";S;" CORRECT","OUT OF A POS
SIBLE ";A;" QUESTIONS";TAB 0;"Y
OUR GRADE AVERAGE IS ";S/A*100;"
PER CENT"

```

```

1180 PAUSE 400
1190 PRINT AT 15,0;"DO YOU WISH
TO SAVE THIS PROGRAM1 = NO",,"0
= YES",B$
1200 LET S=0
1210 INPUT F$
1220 IF CODE F$<=27 OR CODE F$>=
29 THEN GOTO 830
1230 SAVE "QUIZ"
1240 GOTO 830
SYNTACTIC SUM: 40476, 8K ROM

```

NEW FROM SILICON VALLEY!

COMPUWIZ SOFTWARE PRESENTS:

Quality software for your ZX81/TS1000 computer. All at LOW, LOW prices. Super graphics and lots of fun!

1. Blackjack, Hi-Lo.....\$10.95
2. Mastermind, Supermind and Guess.....\$10.95
3. Escape, Snake and The Trap.....\$8.95
4. 2K Game Pack (5 action games).....\$9.95
5. ZX/TS Quick Reference Card.....\$2.50

Games 1, 2 and 3 require 16K.

Send Check or Money Order to:
COMPUWIZ SOFTWARE, Box 390078
Mountain View CA, 94039

Cal. Res. please add 6% tax.
SASE for list of new products.
Dealer inquiries welcomed.

EZRA GROUP II EZRA GROUP II

LOW LOW LOW LOW LOW LOW LOW LOW LOW LOW
PRICES!

Biorhythms 8K ROM/1K&up.....1.00
Graphics Billboard 8/lup.....1.00
TAP WRITER TM(Handicap Aid)8/1.FREE
Horse Race 8/lup.....2.00
SPINNER TM(like Rubik's)8/16...2.00
Improved SLOW PAUSE.....1.00
Linear Regression 8/lup.....2.00
CHAR. Generator Demos 8/lup...2.95
Plotting Work Sheet 8/1.....3.95
Plotting Work Sheet SLOW/16....6.95
TRUTH value PROGRAMMING.....2.00
CHEWTER TM(like PACMN)SLOW/16..2.95
ISLAND SQUARES TM SLOW/2up....2.00
Random MUSIC!SLOW/lup.....2.00
BASIC Keyword Demos 8/1....from .50

ORDER,SASE,reSASE gets you GOO-
DIES Catalog and **FREE** Program

EZRA GROUP II EZRA GROUP II

POB 5222 San Diego,California 92105

Join the CLICK!

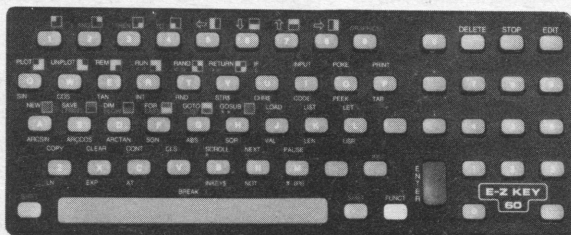
Enter your programs Faster & Easier
with the E-Z Keyboard . . .

At last, a large 60 key "Tactile Feel"
keyboard that plugs into the same
connectors as the existing keyboard
on your ZX81 or your Timex-Sinclair
1000.



Hear the **CLICK** and feel a **SNAP** for every
key pressed! (Tactile Feedback). **Only**

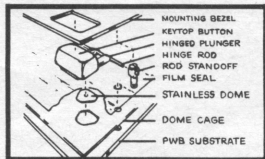
\$84⁹⁵



E-Z Key 60 has the following features:

- 60 Keys - Legends in 3 colors on the base
- Molded legends on key tops
- 8 Automatic shift keys (no shifting required)
for edit, delete, single and double quotes, colon,
semi-colon, function and stop.
- 2 Shift keys - **Numeric key pad** - 5" space bar.

E-Z Key 60 requires no wiring (just plug it in)
and can be adapted to fit the XZ80 or the
MicroAce (8 KROM). The Mounting base
measures 10"x4". Cables and instructions are
included.



SWITCH SPECIFICATIONS:

Key tops measure .4" x .3" - spaced at
3/4" intervals between keys. Life = 10
million operations, typical. Force = 3.
oz. Travel = .040". Dome switch, button
type with arm to give .040" travel.

E-Z KEY

**SUITE 75 A, 711 SOUTHERN ARTERY
QUINCY, MASSACHUSETTS 02169
(617) 773-1187**

A custom made enclosure (shown above) is
also available for your computer and E-Z Key
60 keyboard.

Measurements:	Price:
EC-11 11"x9"x3"	\$25.00
EC-14 14"x9"x3"	\$30.00

WATCH FOR THESE NEW PRODUCTS!

JOYSTICK: Joystick kit that requires no wiring
and will function like the arrow keys & Ø on your
computer.

E-Z Key 40 Replacement flat keyboard with
embossing around each switch and 3 color legends
and graphics as existing keyboard (plug in
replacement).

Delivery 4-6 weeks. 90 day warranty.

USE THIS ORDER FORM...

	Quantity	Unit Price	Total
☐ Check or Money Order	E-Z Key 60	\$84.95	
	EC-11/14	\$25/\$30	
Charge to my:	Total units S&H \$4 per unit		
☐ Visa ☐ Mastercard	Mass. res. add 5% sales tax		
Card #	Total		
Expires			

Send to:

E-Z KEY

Suite 75 A
711 Southern Artery
Quincy, MA 02169

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Build Your Own EPROM Programmer and Centronics Printer Interface – Part II

This is Part II of "Build Your Own EPROM Programmer and Centronics Interface." Part I (SQ, Winter 82) covered building the expansion board and parallel printer port. John Oliger's step-by-step demonstration in Part I also included etching printed circuit boards, printer port modification for use with full 48K RAM, decoding RAM and testing your board. All these projects require at least 16K RAM.

The EPROM (2716) Programmer

Now we are going to build an EPROM (erasable programmable read only memory) programmer and an EPROM read board. The programmer will program only the 2716 or TMS2516 EPROMs and it only programs. You cannot verify with this board; I omitted verify circuitry in the interests of simplicity and cost. An EPROM programmer is not a circuit you use every day. While the ability to verify is a nice convenience, it's not really necessary.

Before you can use the EPROM read board or the EPROM programmer, you must fully decode the ROM to

keep it from conflicting with this new address space. Use the circuit in schematic 6 on one of the spare IC sockets on the expansion board.

If you make the 0-6K reader you will no longer need this circuit. You should remove it to keep it from fighting the decoders on that board.

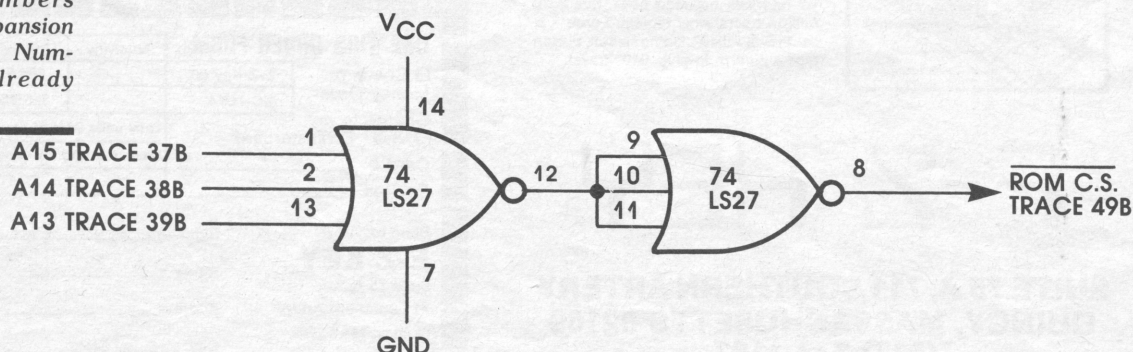
After you program an EPROM, turn off the computer. Remove the programmer board and take the programmed EPROM from it. Install the EPROM in the EPROM read board. Then load your program back into the computer and write a small BASIC program to verify the EPROM.

When you have the printer driver on EPROM with the original program in RAM starting at location 28672, the verification program (listing 5) goes something like this:

Listing 5

```
10 LET X=8192
20 FOR N=28672 TO 28960
30 IF PEEK X<>PEEK N THEN PRIN
T X
40 LET X=X+1
50 NEXT N
```

Schematic 6. All Trace Numbers Refer to Expansion Board Trace Numbers as Already Explained.



Our EPROM programming program (listing 6) goes like this:

Listing 6

```

10 SLOW
20 LET X=8192
30 FOR N=28672 TO 28960
40 POKE X,PEEK N
50 LET X=X+1
60 PAUSE 4
70 NEXT N

```

With my programmer, I've never had a "Not Verify" that was the hardware's fault. The only one I programmed wrong was my own error. If you see dark wide horizontal bars on the screen during programming, the EPROM is in backwards.

The programmer has its own one-shot aboard to time the programming pulse, so you need no software for this. But be careful not to address this block of memory again for at least 50 ms. Hence the PAUSE 4 in line 60. Don't leave this line out!

To build the programmer, etch and plate the board just like the port board you already built. Use board layout 3. On this board you need to jumper from pin 10 of U5 to ground on the bottom. When installing the feedthroughs, be careful not to put any where C6, C5, C1 and R1 mount (just to the left of U3). They are soldered top and bottom, so their leads act like the feedthrough (see schematic 7).

Solder a solid red wire to the bottom of the board to pad marked Vpp. Solder a solid black wire to pad 12 of

the 2716 (ground). These wires should extend about 2 inches above the top of the board. Strip the ends and bend them into an O shape. You will connect the 25 volt supply to these when you program.

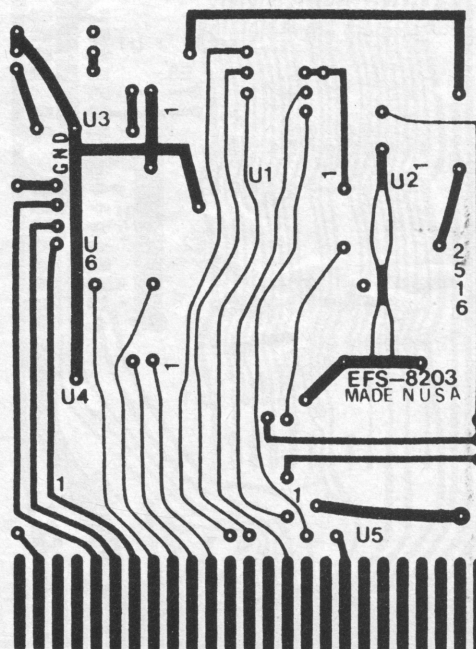
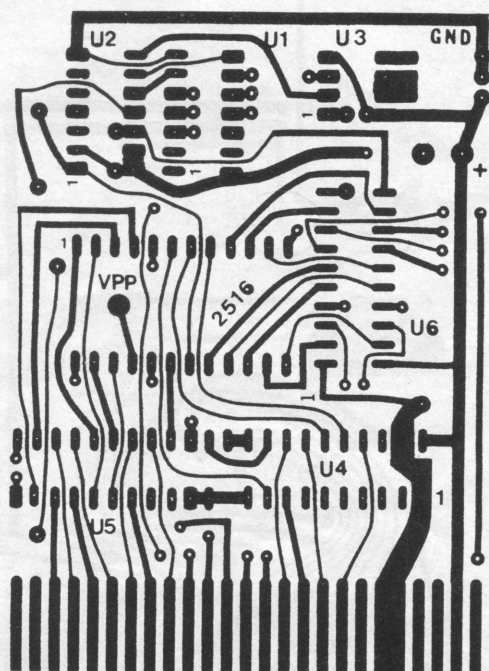
The 25 Vdc regulated power supply necessary to program should have a tolerance of + or - 1 volt. You may use any well filtered and regulated supply you like, or you can construct the circuit in schematic 8 on a small perforated board. It draws very little current, so use the smallest and cheapest transformer you can find.

Turn on the supply just before you run the BASIC program, and turn it off as soon as programming is complete. It should be off while you enter the driver program and while you turn the computer on and off. Without this voltage present, no programming can take place. Use alligator clips on the output wires from your supply to connect it to the solid wires on the board. Be careful to keep them from touching any other boards or parts. Also be careful not to attach them backwards! Use a red alligator clip for +25 and a black one for ground.

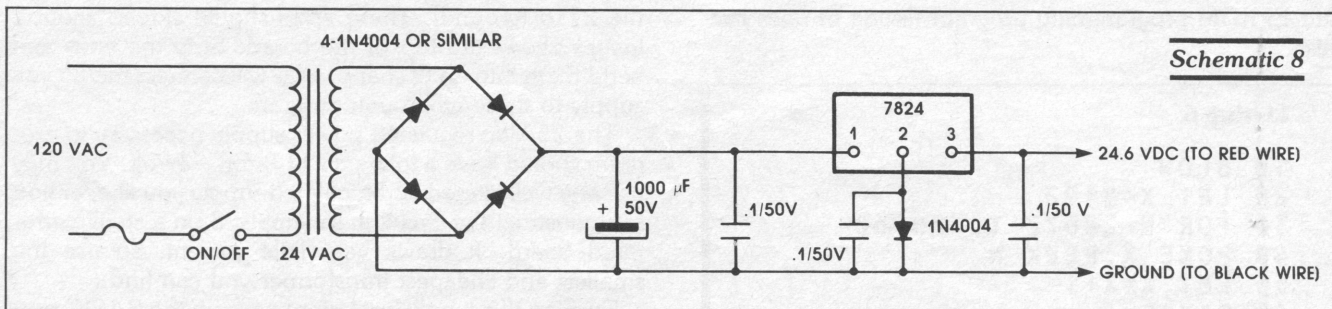
Don't forget to install C2 and C3 on the bottom of the board. All vertical ICs have pin 1 down, while horizontally mounted ICs (U4 and U5) have pin 1 to the left (viewing from component side of board). Because you probably won't program EPROMs for a living, a ZIF (zero insertion force) socket for the EPROM is not necessary. Be sure to use only quality parts for R1 and C1. I used a "monolithic" capacitor for C1 (5%). Use a resistor measured to be very close to 450K.

EPROM Read Boards

After you have the printer converter/driver routine in EPROM, you need to make the 8-16K EPROM read



Layout 3



board. Use the same method we used for all the other boards in these projects, using board layout 4. The socket on the bottom is mapped 8192-10239. Second from the bottom is mapped 10240-12287. Third from the bottom is mapped 12288-14335. The top one is mapped 14336-16383.

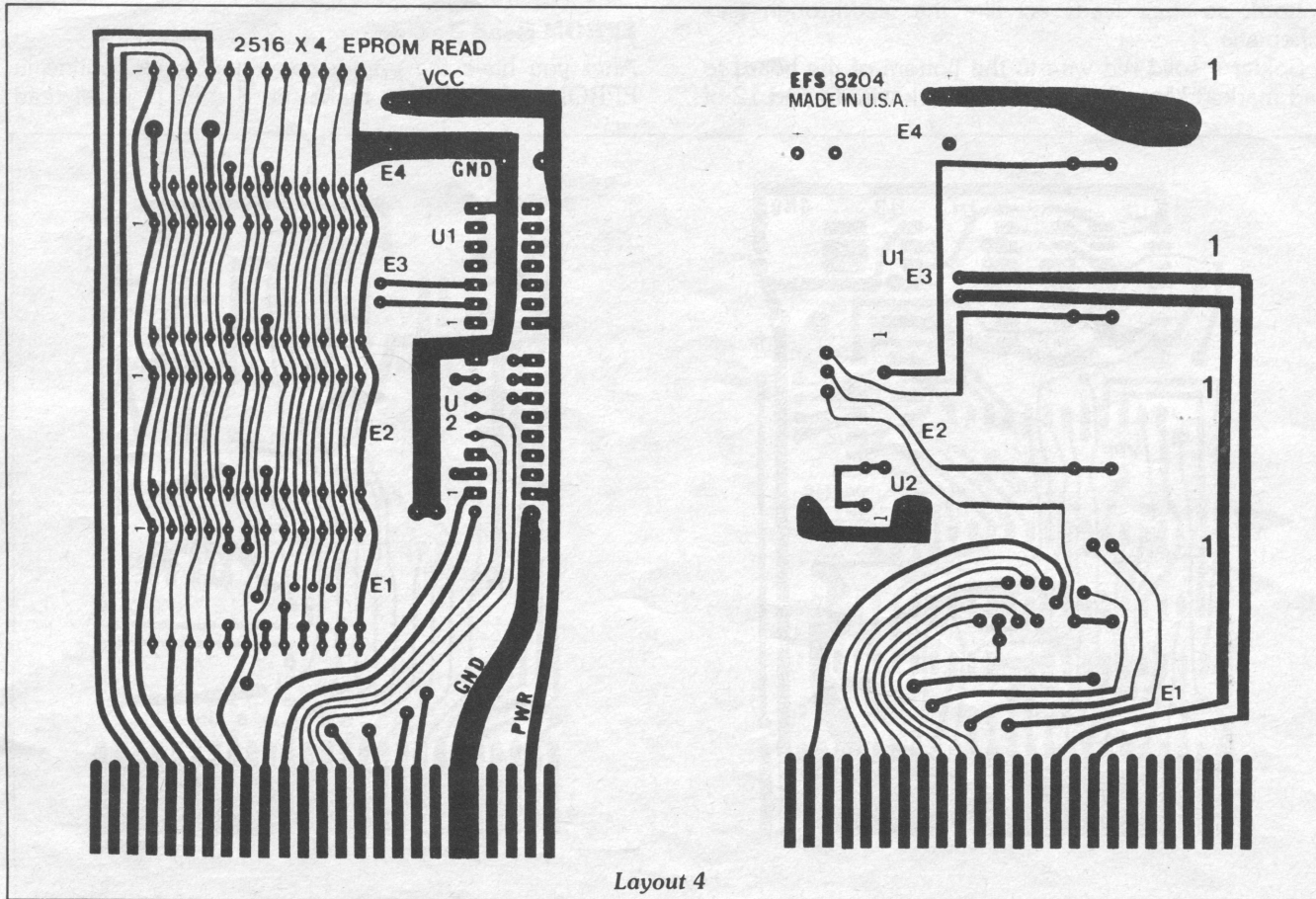
After you build the 8-16K read board, you can burn the printer driver into ROM (listings 9 and 10). Then after you build the 0-6K board, you can use LPRINT, LLIST and COPY to your printer port!

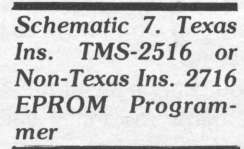
Don't forget to install the wire jumper from pin 1 U1 to pin 6 U2. The tantalum bypass capacitor, C3, should be mounted underneath the bottom EPROM socket. This is because Vcc and ground come into the memory array from the top and are, of necessity, small. The exact value is not critical because it is only really used as a "storage tank." Again, Vcc and ground should be built up with solder for low impedance. C1 is mounted just above U1; C2 is mounted just left of E4. Be sure to observe polarity

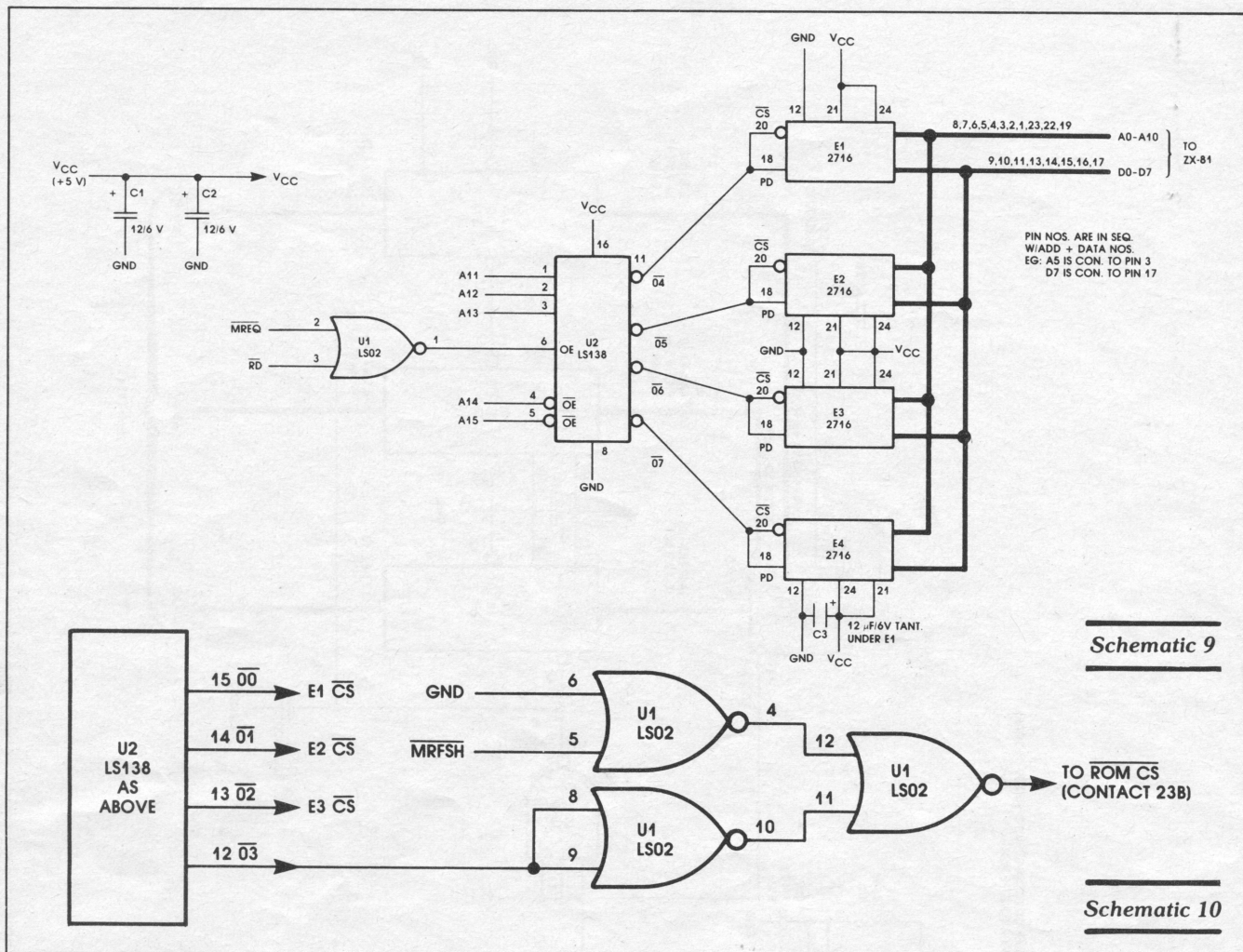
on these components (see schematic 9). If installed backwards, they will sometimes blow up like a firecracker!

Our final project is to construct the 0-6K EPROM read board. On it you will mount one to three EPROMs containing the monitor with a few changes. Construct the 0-6K board just like you are building another 8-16K board, except make the changes listed on the schematic. You will not use the top socket on this board (see schematic 10).

We want at least part of the monitor on EPROM so we can change things. The main changes we want are in the second 2K block (E2—2048-4095), specifically, a call to 02E7 at 0876H. After this patch, we can use all Sinclair's printer commands, LLIST, LPRINT, and COPY. At this point in the ROM, the HL register points at the first character to print and the D register contains the number of lines to print (1 if printing from the printer buffer or 22 if printing from the display). We want to change this call to







02E7 to a call to 20C8. With the call to 20C8, the computer goes to our ASCII convert/drive routine instead of entering the temporary FAST mode as originally intended. Here we save on the stack the registers we will use, convert the characters to ASCII and send them to the printer, get the registers back that we saved, and then simply jump to 02E7 for the monitor. If the ZX printer is attached, it will print the same thing our printer just did. If our printer's own print switch is off, the computer just prints to the ZX printer.

To change 0877 from E7 to C8 and 0878 from 02 to 20, copy this part of the ROM to EPROM with the BASIC program in listing 7 and, of course, our EPROM programmer.

The next program, listing 8, copies the first block of the monitor. In it, we change location 4 from FFH to FEH and location 5 from 7FH to FFH. With this change, the monitor checks all our memory on power up, rather than just 16K. Now if you have greater than 16K on your computer, you won't have to move RAMTOP every time you power up.

You change location 3 to only 254 because the monitor increments the initial value and stores it as RAMTOP. Thus it would increment 255 to 0 and store 0 as RAMTOP. The system would still function, but the display file

Listing 7.

```

10 SLOW
20 LET X=8192
30 FOR N=2048 TO 2166
40 POKE X,PEEK N
50 LET X=X+1
60 PAUSE 4
70 NEXT N
80 PAUSE 4
90 LET X=X+2
100 FOR N=2169 TO 4095
110 POKE X,PEEK N
120 LET X=X+1
130 PAUSE 4
140 NEXT N
150 POKE 8311,200
160 PAUSE 5
170 POKE 8312,32

```

```

10 POKE 8192,PEEK 0
20 PAUSE 5
30 POKE 8193,PEEK 1
40 PAUSE 5
50 POKE 8194,PEEK 2
60 PAUSE 5
70 POKE 8195,254
80 PAUSE 5
90 POKE 8196,255
100 PAUSE 5
110 LET X=8197
120 FOR N=5 TO 2047
130 POKE X,PEEK N
140 LET X=X+1
150 PAUSE 4
160 NEXT N

```

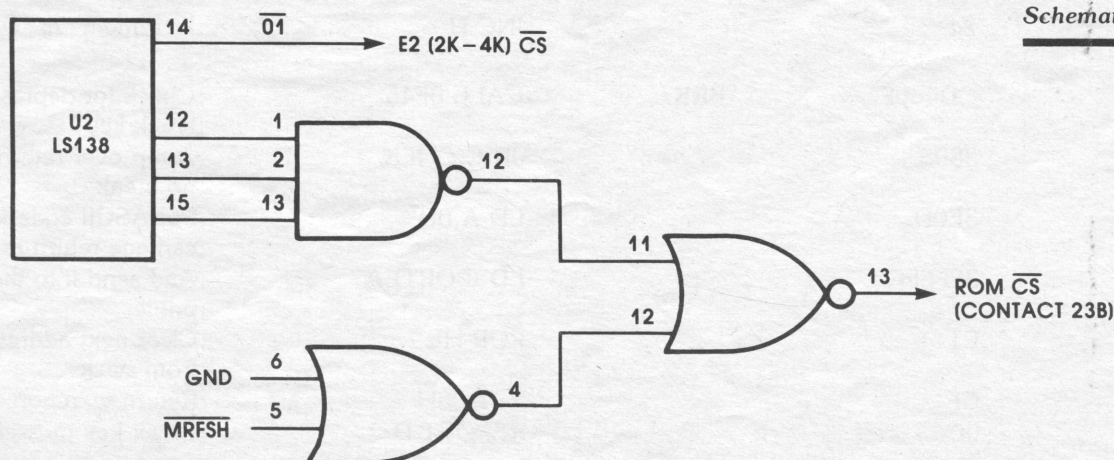
You can program the third block without any changes. However, now that it is on EPROM, you could change it if the need arose. The last block can be programmed directly also if you want to put it on EPROM. If you do, you have to mount it inside the computer and slightly rewire the socket (see below). Because the character generator is in this last block of memory, you may want to put it on EPROM to change the characters. I changed the British pound sign to a "!" because I didn't use the pound sign enough to justify its own key.

ASCII Convert and Copy Routine

ASCII Conversion Table

2000	20	81	82	83	94	95	96	97
2008	FF	FC	83	22	21	24	3A	3F
2010	28	29	3E	3C	3D	2B	2D	2A
2018	2F	3B	2C	2E	30	31	32	33
2020	34	35	36	37	38	39	41	42
2028	43	44	45	46	47	48	49	4A
2030	4B	4C	4D	4E	4F	50	51	52
2038	53	54	55	56	57	58	59	5A
2040	20	20	20	20	20	20	20	20
2048	20	20	20	20	20	20	20	20
2050	20	20	20	20	20	20	20	20
2058	20	20	20	20	20	20	20	20
2060	20	20	20	20	20	20	20	20
2068	20	20	20	20	20	20	20	20
2070	20	20	20	20	20	20	0D	20
2078	20	20	20	20	20	20	20	20
2080	FF	FE	FD	FC	EB	EA	E9	E8
2088	FF	83	FC	40	23	24	7E	25
2090	7B	7D	3E	3C	26	2B	5F	2A
2098	2F	60	27	2E	30	31	32	33
20A0	34	35	36	37	38	39	61	62
20A8	63	64	65	66	67	68	69	6A
20B0	6B	6C	6D	6E	6F	70	71	72
20B8	73	74	75	76	77	78	79	7A

If you wish to put the last block (6-8K) on EPROM, cut the foil trace from pin 18 of the ROM socket and jumper pin 18 to ground (pin 12). Then cut the trace to pin 21 on the ROM socket and jumper pin 21 to Vcc (pin 24). The 6-8K EPROM must be in the main computer because it contains the character generator. Install the programmed 2716 into the ROM socket exactly as the original ROM was installed.

SQ

Schematic 11

Listing 10

<u>Address Location</u>	<u>Contents</u>	<u>Label</u>	<u>Mnemonic</u>	<u>Comment</u>
20C0	1616	COPY	LD D,16	;Set 'D' for 22 lines
20C2	2A0C40		LD HL,(DFILE)	;Get screen memory address
20C5	23		INC HL	;Point at first ;character
20C6	1811		JR CONV	;Go and copy the ;screen
Calls here (20C8) from monitor:				
20C8	3AFFFF	PTCH	LD A,(PORT)	;Get printer status
20CB	CB5F		BIT 3,A	;Is print switch on?
20CD	2807		JR Z RETP	;Return to monitor
20CF	E5		PUSH HL	;if not save registers
20D0	D5		PUSH DE	;used for ZX printer ;use, later
20D1	CDD920		CALL CONV	;Call ASCII Conv/Drive ;Routine
20D4	D1		POP DE	;Restore registers
20D5	E1		POP HL	;for ZX printer use
20D6	C3E702	RETP	JP TFAS	;Jump to 'Temp Fast' ;for monitor
20D9	7E	CONV	LD A,(HL)	;Get character code to ;LPRINT
20DA	23		INC HL	;Point at next ;character
20DB	E5		PUSH HL	;Save location for ;later ;table
20DC	210020		LD HL,TABL	;Point at conversion ;table
20DF	85		ADD L	;Add displacement code
20E0	6F		LD L,A	;Put this in "lo" byte
20E1	3001		JR NC BRK?	;Jump over inc if no ;carry
20E3	24		INC H	;Increment "hi" byte
20E4	CD460F	BRK?	CALL 0F46	;Check for depressed ;break key
20E7	3808		JR C CHEK	;Jump over return if ;no break
20E9	3E0D		LD A,0D	;Put ASCII code for ;carriage return in A
20EB	32FFFF		LD (PORT),A	;And send it to the ;printer
20EE	E1		POP HL	;Clear next address ;from stack
20EF	CF		RST 08H	;Return w/report "D"
20F0	0C		REPORT D	;Break key pressed

Listing 10 Continued

<u>Address Location</u>	<u>Contents</u>	<u>Label</u>	<u>Mnemonic</u>	<u>Comment</u>
20F1	1E02	CHEK	LD E,02	;Set 'E' to check ;status twice
20F3 20F6	3AFFFF FE18	RDY?	LD A,(PORT) CP 18	;Get printer status ;Is it ready to ;receive?
20F8	20EA		JR NZ BRK?	;Loop to break test if ;not
20FA	1D		DEC E	;Decrement check ;counter ;o.k.
20FB	20F6		JR NZ RDY?	;Loop 'till 2 times
20FD	7E	GTIT	LD A,(HL)	;Get converted code to ;send
20FE 2100	FE0D 2012		CP 0D JR NZ SEND	;Is it an "ENTER"? ;Go and send it if not
2102	217B40	LDN?	LD HL,FLAG	;Point at multiline ;flag
2105	F5		PUSH AF	;Save "ENTER" on the ;stack
2106	7E		LD A,(HL)	;Now, get the flag
2107	23		INC HL	;Point at the counter
2108	BE		CP (HL)	;Ready to send "ENTER" ;now?
2109	2805		JR Z CONT	;Clear counter & send ;enter if true
210B 210C 210D	F1 34 15		POP AF INC (HL) DEC D	;Clear the stack ;Increment counter ;Decrement line ;counter
210E	1807		JR 2117	;Jump over driver, ;this time
2110 2111 2113	F1 3600 15	CCNT	POP AF LD (HL),00 DEC D	;Get "ENTER" to send ;Reset the counter ;Decrement the line ;counter
2114	32FFFF	SEND	LD (PORT),A	;Send character to the ;printer
2117	E1		POP HL	;Get next char address ;to convert
2118	20BF		JR NZ CONV	;Go do next char if not ;done
211A	C9		RET	;Return to caller

Operation Codes of the 8080, 8085, and Z80 Processors

D Martin Harrell
313 Hollyberry Rd
Severna Park MD 21146

Manual conversion between assembly language mnemonics and hexadecimal object code can be tedious — particularly if much code is involved. However, the task does not have to be overwhelming. A conversion table helps immensely and is also a good training aid for novice programmers. It presents the entire instruction set in compact form, revealing useful patterns, and also inconsistencies.

8080 and 8085 Operation Codes

Operation codes for the Intel 8080 and 8085 microprocessors are shown in table 1. The only difference between the instruction sets for this pair is that the 8085 has two additional instructions: the read-interrupt-mask instruction (mnemonic RIM, hexadecimal code 20), and the set-interrupt-mask instruction (mnemonic SIM, hexadecimal code 30). They allow the user to control interrupts and a serial I/O (input/output) line, thus making them useful additions.

The position of an 8080/8085 operation code in the table does not give a reliable clue about the implied addressing mode. Table 1 is generally organized according to the operands involved. Residing in the middle eight columns of the table (columns 4 thru B) for example, are the instructions for single-byte move, arithmetic, and logical operations. (Length attributes in this article refer to data, rather than instruction length, unless other-

wise noted.) Regardless of the column, progression through the eight possible choices for the source (second) operand is always in the same sequence as the user moves down a column: registers B thru L; followed by memory reference; and finally, register A, the accumulator. Then, because each column has sixteen entries, the sequence repeats. If the arithmetic and logical instruction groups do not seem to conform to this rule, note that the first operand (always register A) is implied rather than stated explicitly.

This same sequence is used for advancing through choices for the destination (first) operand. In this case, however, progression is column to column from left to right, with each successive column containing two of the eight possible operands. The double-byte instructions also conform to this first-operand type of arrangement. Most of these appear in the first four columns of the table; however, the stack commands to PUSH and POP double-byte data are located at the far right in the top section.

An apparent inconsistency appears in the middle of the table. Hexadecimal code 76 is the instruction to halt the processor (HLT). Expected there instead is MOV M,M, the op code meaning "move the content of the memory location whose address is in the H and L register pair into that

same memory location." The expected instruction is effectively just a slow equivalent of the no operation (NOP) located at hexadecimal 00. Hence, its replacement by the halt command improves, rather than degrades the power of the instruction set. Still, I wonder why an otherwise empty spot in the table was not chosen — as was done for the two additional 8085 instructions.

The right quarter of the table mainly contains program branching and data exchange instructions. Excluding the previously mentioned stack commands, none of these have explicit operands so the previously discussed organization is impossible. The miscellaneous nature of these instructions also tends to prevent predictable order.

Nonetheless, the op codes in this area have a consistent structural style. Most are arranged in complementary order, with mutually exclusive conditions placed in the same column, separated by eight rows. The group of return instructions is typical. The unconditional return from subroutine command is hexadecimal C9. Starting immediately above it and proceeding to the right, four of the eight conditional return instructions are found. The other four (the complements) are eight rows higher.

The order in which these conditions appear is uniform from group to group. To determine that this is so, compare similar elements of the call, jump, and return groups. The unconditional jump (JMP) instruction is a curious exception. Its expected code is CB, but it actually appears eight rows higher in the table. Such exceptions are few enough not to be bothersome.

Z80 Operation Codes

The Z80 is an enhanced version of the 8080. It runs faster, has twice as many general purpose registers, and has a much larger instruction set. Included as a subset in this instruction is the entire repertoire of the 8080. (This compatibility exists at the

machine language level, but not the assembly language level; standard mnemonics and assembly language formats for the two processors differ considerably.) Thus, in hexadecimal object form, almost any program written for the 8080 will produce identical results when executed by a Z80. Because of the Z80's generally higher speed, software timing loops are an exception to this upward compatibility feature. [Editor's note: *There is also a slight difference in the operation of the parity flag. . . .RSS*]

The similarities of the two instruction sets can be seen by comparing corresponding positions of table 1 and table 2. Table 2 is the basic conversion table for the Z80. For every valid 8080 instruction in table 1, its correspondent in table 2 produces logically equivalent results. The differences between the two instruction sets stem from the twelve positions unused by the 8080. These, which are clearly indicated in table 2, are used to greatly expand the Z80's capability.

The Zilog Corp used the seven unfilled positions on the left side of table 1 and the uppermost one on the right side to give the Z80 processor the ability to perform relative branching and to exchange the contents of its two sets of registers. However, the use of hexadecimal codes 20 and 30 for two of the jump relative instructions means that the Z80 is not as compatible with the 8085 as it is with the 8080.

The real expansion of the Z80's instruction set over that of the 8080 is the result of the interesting use of the four other empty spaces in table 1. In essence, the Z80 uses them as pointers to four additional 16 by 16 tables, thus increasing the number of possible op codes by 1532. (The Z80 does not use most of these, but flexibility for future expansion is certainly there.) Had this innovative use of the unimplemented codes not been done, the Z80 would have been limited to 256 different op codes, which is only twelve more than the 8080.

There is a penalty for this flexibility: all instructions in these expansion sets must be multibyte. The first byte identifies the appropriate expansion instruction set, after which, the

second byte identifies the operation to be performed. Sometimes there is an additional third or fourth byte to provide data or addressing information.

Shift, Rotate, and Bit Manipulation Instructions

Consider these pointer instructions one at a time. All of the instructions which begin with hexadecimal CB are contained in table 3. All of the *direct-mode* instructions to shift or rotate (in either direction) any byte in memory or in any of the eight active registers are located here. Table 3 also contains the direct-mode instructions to set, reset, or test any bit in any of these bytes. All of these operations have a length of two bytes. Interestingly, there are more valid instruction combinations derived from the ten basic instructions in this table than there are in the entire 8080 set.

Two features of table 3 are notable. The first is the absence of a "shift left logical" counterpart to the SRL command group. The shift left logical counterpart is not there because it is not needed; the "arithmetic shift left" instructions in column 2 (hexadecimal) accomplish this function. The use of the same general organizational rules indicated earlier for the 8080 is the more important of the two properties of this table. Such uniformity is a good aid in locating instructions in this table.

Indexed Instructions

Instructions beginning with hexadecimal DD are in one of two indexed classes of instructions. These use the IX and IY registers respectively in forming a data address. Those related to the former are depicted in table 4 and its associated table 5.

The analogy between tables 2 and 4 and between tables 3 and 5 is striking. The organizational patterns are identical — even to the point of using the same expansion technique. They should be identical. Each of these indexed instructions was formed by replacing the (HL) operand of an equivalent register-indirect instruction with the indexed notation (IX+d). Thus, every operation that can be performed in the register-indirect mode by the 8080 or Z80 can

also be performed in the indexed mode by the Z80.

The resulting positional equivalence between the two sets of tables is most helpful in determining the required hexadecimal code for the indexed instructions. An easy way to do this without having to refer to tables 4 or 5 is to first select from table 2 or 3 (as appropriate) the hexadecimal code for the register-indirect form of the desired operation. Then place a DD prefix in front of this code if the operation was found in table 2, or a DDCB prefix, if found in table 3. Finally, place after this code group a displacement suffix, d.

The Z80 also has a second index register, which is designated the IY register. Op codes which use it for addressing are contained in tables 6 and 7. It takes only a quick glance to notice the strong similarity between tables 4 and 6 and between tables 5 and 7. As might be expected, virtually everything said previously about the IX class of op codes also refers to the IY class. The sole exception to this statement is that the IY-type instructions begin with hexadecimal code FD, instead of DD.

Miscellaneous Additions

All fifty-six instructions in the last of the four expansion sets begin with hexadecimal code ED. They are listed in table 8. Though they are quite heterogeneous, they add considerably to the power of the Z80. Among these, for example, are instructions that enhance the 16-bit arithmetic capability, set interrupt modes, permit complementing the accumulator, and allow a register-indirect type of I/O to be performed. There are instructions also, which allow counting or block processing to be done during loading, comparison, and I/O operations. Even if the other three expansion sets were omitted, the instructions in this set would be highly useful additions to the basic 8080 complement.

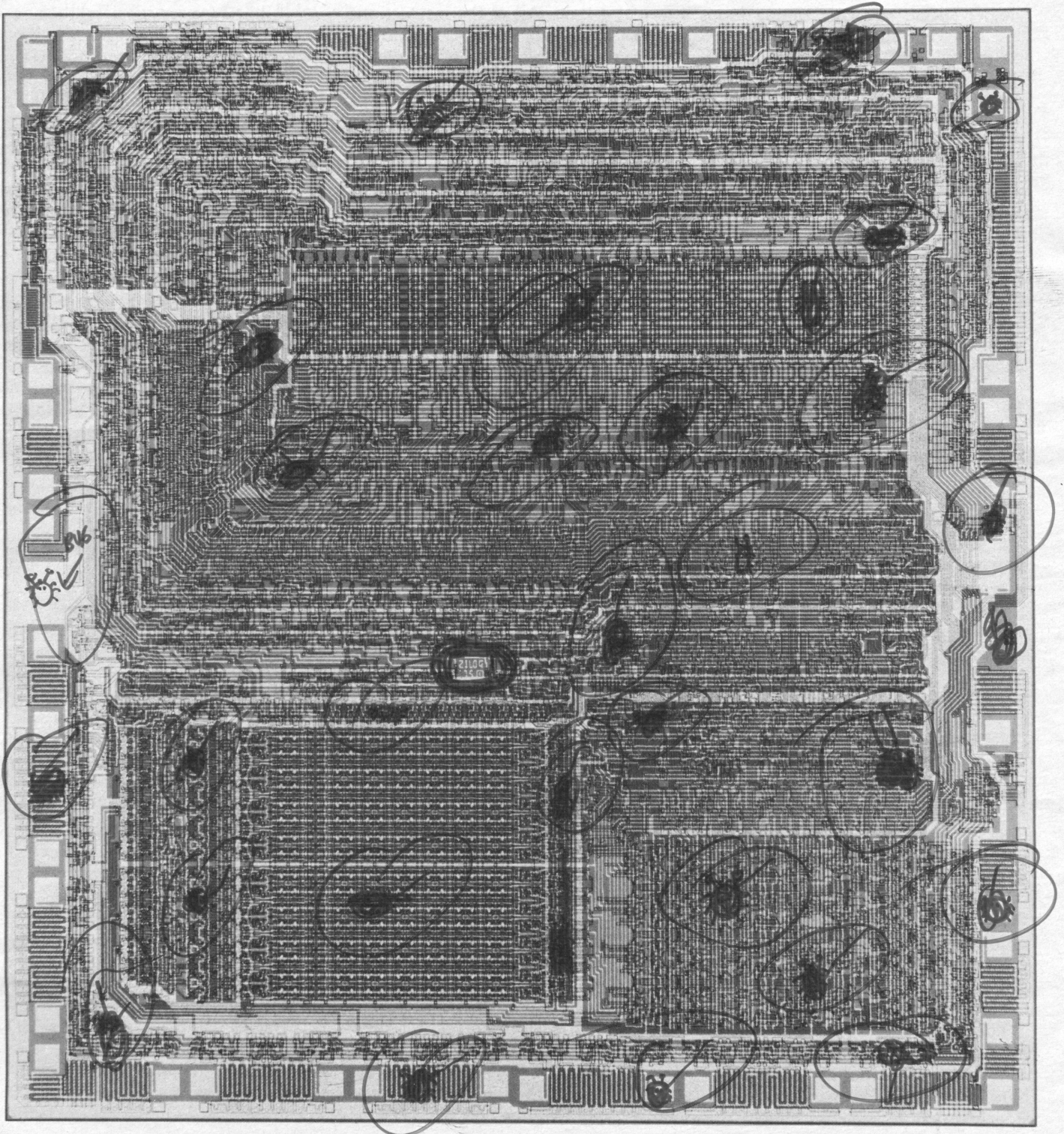
With such a hodgepodge of function, it is rather surprising that any order at all can be made of these ED class instructions. Nonetheless, consistency with the other tables is maintained. It is evident from the arithmetic and the leftmost I/O instruc-

tions that arrangement by order of first and second operands is used whenever possible. Separation of complementary functions by eight rows in a column is also followed.

There are 696 valid op codes in the

seven Z80 tables. Without organizational consistency, conversion of these instructions from mnemonic to hexadecimal form would be extremely difficult and probably ridden with error. Fortunately, these codes are

very well arranged, following the pattern established for the 8080. It takes a little practice to become adept at making these transformations, but with the aid of these tables it can be accomplished successfully. ■



0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP			MOV B,B	MOV D,B	MOV H,B	MOV M,B	ADD B	SUB B	ANA B	ORA B	RNZ	RNC	RPO	RP
1	LXI B	LXI H	LXI SP	MOV B,C	MOV D,C	MOV H,C	MOV M,C	ADD C	SUB C	ANA C	ORA C	POP B	POP D	POP H	POP PSW
2	STAX B	SHLD	STA	MOV B,D	MOV D,D	MOV H,D	MOV M,D	ADD D	SUB D	ANA D	ORA D	JNZ	JNC	JPO	JP
3	INX B	INX H	INX SP	MOV B,E	MOV D,E	MOV H,E	MOV M,E	ADD E	SUB E	ANA E	ORA E	JMP	OUT	XTHL	DI
4	INR B	INR H	INR M	MOV B,H	MOV D,H	MOV H,H	MOV M,H	ADD H	SUB H	ANA H	ORA H	CNZ	CNC	CPO	CP
5	DCR B	DCR H	DCR M	MOV B,L	MOV D,L	MOV H,L	MOV M,L	ADD L	SUB L	ANA L	ORA L	PUSH B	PUSH D	PUSH H	PUSH PSW
6	MVI B	MVI H	MVI M	MOV B,M	MOV D,M	MOV H,M	HLT	ADD M	SUB M	ANA M	ORA M	ADI	SUI	ANI	ORI
7	RLC	RAL	DAA	MOV B,A	MOV D,A	MOV H,A	MOV M,A	ADD A	SUB A	ANA A	ORA A	RST 0	RST 16	RST 32	RST 48
8				MOV C,B	MOV E,B	MOV L,B	MOV A,B	ADC B	SBB B	XRA B	CMP B	RZ	RC	RPE	RM
9	DAD B	DAD H	DAD SP	MOV C,C	MOV E,C	MOV L,C	MOV A,C	ADC C	SBB C	XRA C	CMP C	RET		PCHL	SPHL
A	LDAX B	LDAX D	LHLD	MOV C,D	MOV E,D	MOV L,D	MOV A,D	ADC D	SBB D	XRA D	CMP D	JZ	JC	JPE	JM
B	DCX B	DCX D	DCX H	MOV C,E	MOV E,E	MOV L,E	MOV A,E	ADC E	SBB E	XRA E	CMP E		IN	XCHG	EI
C	INR C	INR E	INR L	MOV C,H	MOV E,H	MOV L,H	MOV A,H	ADC H	SBB H	XRA H	CMP H	CZ	CC	CPE	CM
D	DCR C	DCR E	DCR L	MOV C,L	MOV E,L	MOV L,L	MOV A,L	ADC L	SBB L	XRA L	CMP L	CALL			
E	MVI C	MVI E	MVI L	MOV C,M	MOV E,M	MOV L,M	MOV A,M	ADC M	SBB M	XRA M	CMP M	ACI	SBI	XRI	CPI
F	RRC	RAR	CMA	MOV C,A	MOV E,A	MOV L,A	MOV A,A	ADC A	SBB A	XRA A	CMP A	RST 8	RST 24	RST 40	RST 56

Second Nibble

Table 1: Mnemonics of the operation codes of the 8080 and 8085 microprocessors arranged conveniently for conversion to the hexadecimal object code. This task is aided by the organizational consistency of the instruction set. The two instructions (RIM and SIM) found only in the 8085 are indicated by shading.

First Nibble

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	JR NZ,e	JR NC,e	LD B,B	LD D,B	LD H,B	LD (HL),B	ADD A,B	SUB B	AND B	OR B	RET NZ	RET NC	RET PO	RET P
1	LD BC,nn	LD HL,nn	LD SP,nn	LD B,C	LD D,C	LD H,C	LD (HL),C	ADD A,C	SUB C	AND C	OR C	POP BC	POP DE	POP HL	POP AF
2	LD (BC),A	LD (DE),A	LD (nn),HL	LD B,D	LD D,D	LD H,D	LD (HL),D	ADD A,D	SUB D	AND D	OR D	JP NZ,nn	JP NC,nn	JP PO,nn	JP P,nn
3	INC BC	INC DE	INC HL	LD B,E	LD D,E	LD H,E	LD (HL),E	ADD A,E	SUB E	AND E	OR E	JP nn	OUT (n),A	EX (SP),HL	DI
4	INC B	INC D	INC H	LD B,H	LD D,H	LD H,H	LD (HL),H	ADD A,H	SUB H	AND H	OR H	CALL NZ,nn	CALL NC,nn	CALL PO,nn	CALL P,nn
5	DEC B	DEC D	DEC H	LD B,L	LD D,L	LD H,L	LD (HL),L	ADD A,L	SUB L	AND L	OR L	PUSH BC	PUSH DE	PUSH HL	PUSH AF
6	LD B,n	LD D,n	LD (HL),n	LD B,(HL)	LD D,(HL)	LD H,(HL)	HALT	ADD A,(HL)	SUB (HL)	AND (HL)	OR (HL)	ADD A,n	SUB n	AND n	OR n
7	RJCA	RJA	DAA	LD B,A	LD D,A	LD H,A	LD (HL),A	ADD A,A	SUB A	AND A	OR A	RST 00H	RST 10H	RST 20H	RST 30H
8	EX AF,AF	JR e	JR C,e	LD C,B	LD E,B	LD L,B	LD A,B	ADC A,B	SBC A,B	XOR B	CP B	RET Z	RET C	RET PE	RET M
9	ADD HL,DE	ADD HL,HL	ADD HL,SP	LD C,C	LD E,C	LD L,C	LD A,C	ADC A,C	SBC A,C	XOR C	CP C	RET	EXX	JP (HL)	LD SP,HL
A	LD A,(BC)	LD A,(DE)	LD HL,(nn)	LD C,D	LD E,D	LD L,D	LD A,D	ADC A,D	SBC A,D	XOR D	CP D	JP Z,nn	JP C,nn	JP PE,nn	JP M,nn
B	DEC BC	DEC DE	DEC HL	LD C,E	LD E,E	LD L,E	LD A,E	ADC A,E	SBC A,E	XOR E	CP E	See Table 3	IN A,(n)	EX DE,HL	EI
C	INC C	INC E	INC L	LD C,H	LD E,H	LD L,H	LD A,H	ADC A,H	SBC A,H	XOR H	CP H	CALL Z,nn	CALL C,nn	CALL PE,nn	CALL M,nn
D	DEC C	DEC E	DEC L	LD C,L	LD E,L	LD L,L	LD A,L	ADC A,L	SBC A,L	XOR L	CP L	CALL nn	See Table 4	See Table 8	See Table 6
E	LD C,n	LD E,n	LD L,n	LD C,(HL)	LD E,(HL)	LD L,(HL)	LD A,(HL)	ADC A,(HL)	SBC A,(HL)	XOR (HL)	CP (HL)	ADC A,n	SBC A,n	XOR n	CP n
F	RJCA	RJA	CGF	LD C,A	LD E,A	LD L,A	LD A,A	ADC A,A	SBC A,A	XOR A	CP A	RST 08H	RST 18H	RST 28H	RST 38H

Second Nibble

Table 2: Mnemonics of the operation codes of the Z80 microprocessor arranged for conversion to hexadecimal object code. Corresponding positions of table 1 and table 2 generally perform the identical function, despite differences in notation. Enhancements of the 8080 instruction set are indicated by shading. Mnemonics used here are those specified by Zilog.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
RLC B	RL B	SIA B		BIT 0,B	BIT 2,B	BIT 4,B	BIT 6,B	RES 0,B	RES 2,B	RES 4,B	RES 6,B	SET 0,B	SET 2,B	SET 4,B	SET 6,B
RLC C	RL C	SIA C		BIT 0,C	BIT 2,C	BIT 4,C	BIT 6,C	RES 0,C	RES 2,C	RES 4,C	RES 6,C	SET 0,C	SET 2,C	SET 4,C	SET 6,C
RLC D	RL D	SIA D		BIT 0,D	BIT 2,D	BIT 4,D	BIT 6,D	RES 0,D	RES 2,D	RES 4,D	RES 6,D	SET 0,D	SET 2,D	SET 4,D	SET 6,D
RLC E	RL E	SIA E		BIT 0,E	BIT 2,E	BIT 4,E	BIT 6,E	RES 0,E	RES 2,E	RES 4,E	RES 6,E	SET 0,E	SET 2,E	SET 4,E	SET 6,E
RLC H	RL H	SIA H		BIT 0,H	BIT 2,H	BIT 4,H	BIT 6,H	RES 0,H	RES 2,H	RES 4,H	RES 6,H	SET 0,H	SET 2,H	SET 4,H	SET 6,H
RLC L	RL L	SIA L		BIT 0,L	BIT 2,L	BIT 4,L	BIT 6,L	RES 0,L	RES 2,L	RES 4,L	RES 6,L	SET 0,L	SET 2,L	SET 4,L	SET 6,L
RLC (HL)	RL (HL)	SIA (HL)		BIT 0,(HL)	BIT 2,(HL)	BIT 4,(HL)	BIT 6,(HL)	RES 0,(HL)	RES 2,(HL)	RES 4,(HL)	RES 6,(HL)	SET 0,(HL)	SET 2,(HL)	SET 4,(HL)	SET 6,(HL)
RLC A	RL A	SIA A		BIT 0,A	BIT 2,A	BIT 4,A	BIT 6,A	RES 0,A	RES 2,A	RES 4,A	RES 6,A	SET 0,A	SET 2,A	SET 4,A	SET 6,A
RRC B	RR B	SRA B	SRL B	BIT 1,B	BIT 3,B	BIT 5,B	BIT 7,B	RES 1,B	RES 3,B	RES 5,B	RES 7,B	SET 1,B	SET 3,B	SET 5,B	SET 7,B
RRC C	RR C	SRA C	SRL C	BIT 1,C	BIT 3,C	BIT 5,C	BIT 7,C	RES 1,C	RES 3,C	RES 5,C	RES 7,C	SET 1,C	SET 3,C	SET 5,C	SET 7,C
RRC D	RR D	SRA D	SRL D	BIT 1,D	BIT 3,D	BIT 5,D	BIT 7,D	RES 1,D	RES 3,D	RES 5,D	RES 7,D	SET 1,D	SET 3,D	SET 5,D	SET 7,D
RRC E	RR E	SRA E	SRL E	BIT 1,E	BIT 3,E	BIT 5,E	BIT 7,E	RES 1,E	RES 3,E	RES 5,E	RES 7,E	SET 1,E	SET 3,E	SET 5,E	SET 7,E
RRC H	RR H	SRA H	SRL H	BIT 1,H	BIT 3,H	BIT 5,H	BIT 7,H	RES 1,H	RES 3,H	RES 5,H	RES 7,H	SET 1,H	SET 3,H	SET 5,H	SET 7,H
RRC L	RR L	SRA L	SRL L	BIT 1,L	BIT 3,L	BIT 5,L	BIT 7,L	RES 1,L	RES 3,L	RES 5,L	RES 7,L	SET 1,L	SET 3,L	SET 5,L	SET 7,L
RRC (HL)	RR (HL)	SRA (HL)	SRL (HL)	BIT 1,(HL)	BIT 3,(HL)	BIT 5,(HL)	BIT 7,(HL)	RES 1,(HL)	RES 3,(HL)	RES 5,(HL)	RES 7,(HL)	SET 1,(HL)	SET 3,(HL)	SET 5,(HL)	SET 7,(HL)
RRC A	RR A	SRA A	SRL A	BIT 1,A	BIT 3,A	BIT 5,A	BIT 7,A	RES 1,A	RES 3,A	RES 5,A	RES 7,A	SET 1,A	SET 3,A	SET 5,A	SET 7,A

Table 3: Enhancement operation codes of the Z80 invoked by the hexadecimal CB instruction prefix. These CB class operations give bit manipulation, data shifting, and enhanced rotation capability to the Z80.

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
							LD (IX+d),B								
		LD IX,nn					LD (IX+d),C							POP IX	
		LD (nn),IX					LD (IX+d),D								
		INC IX					LD (IX+d),E							EX (SP),IX	
			INC (IX+d)				LD (IX+d),H								
			DEC (IX+d)				LD (IX+d),L							PUSH IX	
			LD (IX+d),n	LD B,(IX+d)	LD D,(IX+d)	LD H,(IX+d)		ADD A,(IX+d)	SUB (IX+d)	AND (IX+d)	OR (IX+d)				
							LD (IX+d),A								
ADD IX,BC	ADD IX,DE	ADD IX,IX												JP (IX)	LD SP,IX
A		LD IX,(nn)													
B		DEC IX													
C															
D															
E				LD C,(IX+d)	LD E,(IX+d)	LD L,(IX+d)	LD A,(IX+d)	ADC A,(IX+d)	SBC A,(IX+d)	XOR (IX+d)	CP (IX+d)				
F															

Table 4: Operations of the Z80 invoked by the instruction prefix DD. These provide indexed-mode instructions equivalent to the indirect-mode instructions and employ the IX register.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6	RLC (IX+d)	RL (IX+d)	SLA (IX+d)		BIT 0, (IX+d)	BIT 2, (IX+d)	BIT 4, (IX+d)	BIT 6, (IX+d)	RES 0, (IX+d)	RES 2, (IX+d)	RES 4, (IX+d)	RES 6, (IX+d)	SET 0, (IX+d)	SET 2, (IX+d)	SET 4, (IX+d)	SET 6, (IX+d)
7																
8																
9																
A																
B																
C																
D																
E	RRC (IX+d)	RR (IX+d)	SRA (IX+d)	SHL (IX+d)	BIT 1, (IX+d)	BIT 3, (IX+d)	BIT 5, (IX+d)	BIT 7, (IX+d)	RES 1, (IX+d)	RES 3, (IX+d)	RES 5, (IX+d)	RES 7, (IX+d)	SET 1, (IX+d)	SET 3, (IX+d)	SET 5, (IX+d)	SET 7, (IX+d)
F																

Table 5: These DDCB-class operation codes are an indexed equivalent of the indirect-mode operation codes of the Z80 shown in table 3.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								LD (IX+d), B								
1			LD IX, nn					LD (IX+d), C							POP IX	
2			LD (nn), IX					LD (IX+d), D								
3			INC IX					LD (IX+d), E							EX (SP), IX	
4				INC (IX+d)				LD (IX+d), H								
5				DEC (IX+d)				LD (IX+d), L							PUSH IX	
6				LD (IX+d), n	LD B, (IX+d)	LD D, (IX+d)	LD H, (IX+d)		ADD A, (IX+d)	SUB (IX+d)	AND (IX+d)	OR (IX+d)				
7								LD (IX+d), A								
8																
9	ADD IX, BC	ADD IX, DE	ADD IX, IX	ADD IX, SP											JP (IX)	LD SP, IX
A			LD IX, (nn)													
B			DEC IX													
C													See Table 7			
D																
E					LD C, (IX+d)	LD E, (IX+d)	LD L, (IX+d)	LD A, (IX+d)	ADC A, (IX+d)	SBC A, (IX+d)	XOR (IX+d)	CP (IX+d)				
F																

Table 6: Indexed instructions employing the IX register. Note the similarity with table 4. These operation codes begin with the FD prefix.

CHANGE YOUR TIMEX/SINCLAIR 1000 DISPLAY TO FULL COLOUR GRAPHICS with **KOLORWORKS**

LOOK AT THE FEATURES !!!

- ♦ Plugs into ZX81/1000 (edge connector)
- ♦ User defined characters & graphics up to 256x192 pixels
- ♦ Latest technology with TMS9918 VDP (32 sprite levels)
- ♦ Module contains extension of basic commands including: PAPER/INK/BORDER/BIN/SPRITE/OUT/INP/etc.
- ♦ Module contains it's own memory
- ♦ All text will run on the color tv

for \$149.95

**KOLORWORKS COMES WITH A LIMITED WARRANTY ON PARTS AND WORKMANSHIP
USE YOUR KOLORWORKS IMMEDIATELY WITH A GAME CASSETTE FOR \$9.95**

This delightful game is designed for hours of fun using some of the color graphic capabilities of KOLORWORKS. The cassette also contains a short program to familiarize you with some of the commands and graphics.

Enjoy Game Fun With

GAAMWORKS

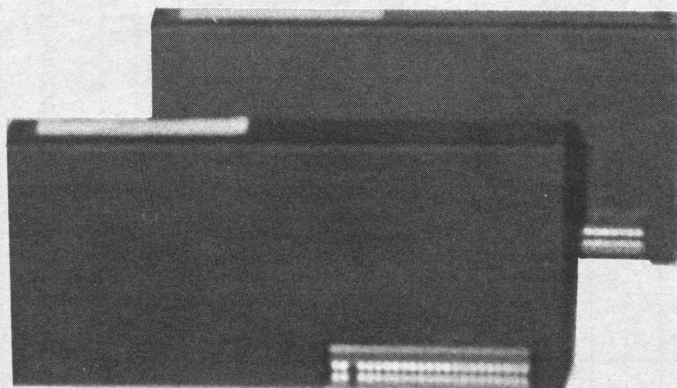
THE SOON TO BE RELEASED GAME MODULE (proto-type stage) WILL OFFER SOUND, ROM CARTRIDGES AND JOY STICK PORTS FOR YOUR TS1000/ZX81.

- ♦ **THE SOUND** will be of arcade game quality which you can program for music, animals, transportation (auto, train, airplane, etc.) and machine sounds.
- ♦ **ROM CARTRIDGES** will have up to 8K of ROM using either 2716, 2732 or 2764 EPROMS. We will have pre-programed cartridges and blank cartridges which you can program. We will be offering a service to burn EPROMS from your cassettes.
- ♦ **JOY STICK PORTS** will allow for the use of two "Atari" compatible joy sticks.

SORRY PRICE IS NOT AVAILABLE AT THIS TIME! GAAMWORKS WILL BE AVAILABLE BY MAIL ORDER STARTING MARCH 1st. FOR FURTHER INFORMATION, SEND \$2.00 (Credited to Order).

At this time KOLORWORKS and GAAMWORKS is available only by mail order.

MAIL TO: **BRAINCHILD COMPUTER WORKS, INC.**
P.O. Box 506
Pewaukee, WI 53072



	PRICE	QTY.	AMOUNT
KOLORWORKS	\$149.95		
CASSETTE	9.95		
Shipping & Handling	4.95		4.95
Wi. residents add sales tax	TAX		
	TOTAL		
	ENCLOSED		

My ☐ check ☐ money order is enclosed

Name _____

Street _____

City _____ State _____ Zip _____

Please allow six to eight weeks for processing. Thank you.

First Nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6	RLC (IY+d)	RL (IY+d)	SLA (IY+d)		BIT 0, (IY+d)	BIT 2, (IY+d)	BIT 4, (IY+d)	BIT 6, (IY+d)	RES 0, (IY+d)	RES 2, (IY+d)	RES 4, (IY+d)	RES 6, (IY+d)	SET 0, (IY+d)	SET 2, (IY+d)	SET 4, (IY+d)	SET 6, (IY+d)
7																
8																
9																
A																
B																
C																
D																
E	RRC (IY+d)	RR (IY+d)	SRA (IY+d)	SRL (IY+d)	BIT 1, (IY+d)	BIT 3, (IY+d)	BIT 5, (IY+d)	BIT 7, (IY+d)	RES 1, (IY+d)	RES 3, (IY+d)	RES 5, (IY+d)	RES 7, (IY+d)	SET 1, (IY+d)	SET 3, (IY+d)	SET 5, (IY+d)	SET 7, (IY+d)
F																

Second Nibble

Table 7: Indexed instructions of the FDCB class, again employing the IY register. Note the similarity with table 5.

First Nibble

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					IN B, (C)	IN D, (C)	IN H, (C)				LDI	LDIR				
1					OUT (C), B	OUT (C), D	OUT (C), H				CPI	CPDR				
2					SBC HL, BC	SBC HL, DE	SBC HL, HL	SBC HL, SP			IN I	INIR				
3					LD (nn), BC	LD (nn), DE		LD (nn), SP			OUTI	OTIR				
4					NEG											
5					RETN											
6					IN 0	IN 1										
7					LD I, A	LD A, I	RHD									
8					IN C, (C)	IN E, (C)	IN L, (C)	IN A, (C)			LDD	LDLR				
9					OUT (C), C	OUT (C), E	OUT (C), L	OUT (C), A			CPD	CPDR				
A					ADC HL, BC	ADC HL, DE	ADC HL, HL	ADC HL, SP			IND	INDR				
B					LD BC, (nn)	LD DE, (nn)		LD SP, (nn)			OUTD	OTDR				
C																
D					RETI											
E						IN 2										
F					LD P, A	LD A, R	RLD									

Second Nibble

Table 8: The class of miscellaneous instructions invoked by the ED prefix.



Sky-High ZX/TS

Name: Flight Simulation
 Type: Game/Education
 ROM/RAM reqd: 8K/16K
 Listable? Yes
 Printed listing? No
 Screen prompts? Yes
 Easy to load? Yes
 Display: Excellent
 From: Sinclair Research Ltd.
 3 Sinclair Plaza
 Nashua, NH 03061
 Price: \$9.95

Psion's Flight Simulation comes on tape with a seven-page instruction leaflet that fits inside the cassette case. The detailed instructions provide a good introduction to flying novices. After a short time, you'll have no trouble remembering what keys control which aircraft functions. Instead, you can concentrate on flying and landing safely.

My tape consistently loads for me every time I use it. The "flip" side is blank. The program is listable and includes a good bit of machine code.

Flight Simulation sets up two situations. You may choose just the final approach or "full feature." The final approach situation lines you up nicely on the correct vector about five miles out, 800 feet in the air. Full feature sets you randomly within 25 miles of the airport. You must pilot yourself to the correct vector and altitude for your final approach. It is not as easy as it sounds, especially if you allow the wind to be a factor.

You get three different screen displays—cockpit, map or moving runway.

On the cockpit display, you see the pilot's view of the instruments and a silhouette of the horizon as seen from inside the aircraft. Given the ZX/TS graphics, it is a very effective, functional display.

For a map of the entire area around the airport, use the map display. The map extends about 20 to 25 miles from the landing field. Shown on the map are the landing field, the six beacons you can use for guidance, some cliffs east of the landing strip, and your heading. A flashing dot on the map represents your current position. This very nice display is remarkably detailed.

Display three is a moving display of the runway as you approach for landing. Yes, it does move. It also shows your speed and altitude. This

display is extremely well done and impressive to see.

You may crash your craft in a variety of ways, with a nice graphic explosion and subsequent crash report telling you what happened. You may tear off your wing flaps, land with your landing gear up, smack into the escarpment at too low an altitude, go into irreversible dives, and more (all of which I've done).

This is the most impressive game program I've seen for the ZX/TS. It's definitely not a make-do-with-what-I've-got program. Rather, it uses the ZX/TS capabilities well to turn out an excellent simulation. This program is a terrific bargain at \$9.95. Thanks to Psion for making Flight Simulation, and to Sinclair for providing it in the US.

Michael Roberts
 Des Moines, IA

New Calc

Name: VU-CALC
 Type: Business/Financial
 ROM/RAM required: 8K/16K
 (ZX81 or
 TS1000 only)

From: Sinclair Research Ltd.
 3 Sinclair Plaza
 Nashua, NH 03061
 or
 Stanhope Road
 Camberley, Surrey
 England GU15 3PS

Written by: Psion Software
 Price: \$14.95 or £7.95

VU-CALC is a program to calculate and display tables of numbers and names. The basic table consists of a grid—26 rows labelled A to Z and 36 columns labelled 01 to 36. The grid size is fixed, so you must load and save the entire grid, whether you need it or not.

Loading a VU-CALC tape with no data takes about two and a half minutes. Once you enter even one number, the program dimensions the grid. To save and load a program containing any amount of data takes six minutes.

This grid is displayed in segments of nine rows and three columns (see Figure 1). Fortunately, scrolling is rapid so you can access any grid position quickly. VU-CALC treats titles as data and scrolls them off the screen. But you quickly get used to this and learn to equate grid designations with titles.

For each grid position you want the computer to calculate, you must provide a formula. A formula consists of grid designations, constants if necessary, and any of the arithmetic operators + (add), - (subtract), * (multiply) and / (divide). Each formula can contain a maximum of 32 characters. Since each grid designation requires three characters, you can add together a maximum of eight rows or columns, for example, A02 + B02 + C02 + D02 + E02 + F02 + G02 + H02. Fortunately you don't need to type this lengthy formula at each grid position you need to compute. VU-CALC can repeat a formula along a row or column. It can be repeated absolutely (exactly as entered) or relatively (with the grid designations incremented appropriately). Sinclair's instruction sheet states simply, "You may use up to 40

	F=FORMULA	L=DATA	C=CALCULATE
	01	02	03
A			
B			
C			
D			
E			
F			
G			
H			
I			

FIG. 1

formulae" and here is where a serious program deficiency arises.

I first interpreted this statement to mean you can only calculate 40 grid positions, but this proved incorrect. Apparently it means you can enter 40 different formulae. A formula repeated 36 times along a row counts as only one formula. Although you can delete a formula from a given grid position, you can never delete a formula from the subroutine that checks for the 40-formula limit. This means you will

eventually reach an unrecoverable error condition where you can enter no more formulae. You can get there very quickly if you use formulae to enter data into your table. You cannot know how many formulae you have entered (short of counting), so the error comes without warning. I have written to Sinclair asking how to recover from this error, but they have not yet answered.

Numbers entered in the grid are aligned along the left side, not along the decimal point. You can enter up

to eight numbers in each grid position. Calculation attempted on a grid containing an alphabetic character or an empty grid position produces a C/5110 error. Load empty grid positions with zeros if you wish to include them in calculations.

Instructions supplied by Sinclair consist of seven sheets sized to fit in the tape box. Rather than teach you how to use the program, they give you just enough information to let you experiment and teach yourself.

VU-CALC is written in machine code and runs adequately fast except when you enter formulae or data. You must pause between each keystroke when entering data to keep it from getting lost. Possibly this delay is not noticeable with the standard ZX/TS keyboard, but anyone with a typewriter-style keyboard will certainly notice it. A numeric keypad would be useless for entering data due to the program's slow response.

Although VU-CALC has several shortcomings, it is a very useful tool for anyone who has to work with tables of numbers. If Sinclair or Psion can resolve the "over 40 formulae" error condition, I will give VU-CALC a high recommendation.

D.V. Carville
APO New York

Magic on Tape

Name: Programmers' Toolkit;
Graphics Toolkit

Type: Utility

ROM/RAM reqd: 8K/16K

Written in: Machine code

Listable? No

Printed listing? No

Easy to load? Yes

From: Softsync Inc.

14 E. 34th St.

New York, NY 10016

212/685-2080

Price: \$14.95 each, 2/\$25 mail
order only

Programmers' and Graphics Toolkits are two separate utility packages. Together they form a dynamite package to help you write professional-looking programs for your ZX/TS computer.

These toolkits let you store machine code routines in your computer to perform often-used tasks or tricks neatly and quickly. You load each in the usual way (one requires an initial POKE). They move themselves above RAMTOP, the place the computer thinks is the end of memory. Then you can enter or load other BASIC programs as if the machine code routines were not even there. The machine code programs are protected against being overwritten by BASIC programs. But whenever you need a toolkit routine, just enter the appropriate USR call and it executes.

You can use each toolkit separately or both together; they are compatible. Each comes with cards giving loading tips, directions, command summaries, and examples for you to try out. Be careful when

following the directions. I found three errors in the instructions for the first graphics routine. They are simple to find and fix; two are left-out greater than and less than signs, the third is an extra character in a line number (105 should be 15).

One particularly good feature is the Graphics Toolkit's error messages. If you make a mistake using a routine you aren't likely to crash the machine. Instead you get a report code akin to the computer's own, telling you the error and the line number it occurred in. Toolkit error codes use letters G to S, so they don't coincide with the computer's own error codes.

Programmers' Toolkit offers these routines: line renumbering (also renumbers GOTOs and GOSUBs), search and replace, search and list, hypergraphics mode (alters the

starting address of the character set), screen fill, reverse video, wait (programs holds until the computer receives input from the tape recorder) and bytes free.

Graphics Toolkit offers 23 routines in all: draw/undraw (define your own shapes), foreground on/off, border/unborder, screen fill, reverse video, up/down/left/right (alter next PRINT position), editprint (changes current PRINT position to screen line 23), upscroll, downscroll, rightscroll, leftscroll, onscreen/offscreen (turns screen display on and off), background on/off, search and replace, and square (draws a square or rectangle to specified coordinates).

Each program requires a 16K RAM pack because of the RAMTOP adjustment. Using the two toolkits together, you still have about 13.5K of memory available.

With the USR calls to the machine code routines, you effectively add 32 commands to the Timex Sinclair BASIC command set. Although the instruction cards are concise, they offer no one-location summary of all the USR commands.

These two tapes can give you the memory conservation and quickness of machine code routines without the aggravation of learning to program yourself or the drudgery of POKEing in someone else's machine code. I really enjoyed the "magic" I could call up with a few keystrokes.

Go FORTH

Name: ZXFORTH
Type: Language
ROM/RAM required: 8K/16K
Written in: Machine Code
Listable? No
Easy to load? Moderately
Printed listing? No
Documentation: Poor
By: International Publishing & Software Inc.

From: Gladstone Electronics
1585 Kenmore Ave.
Buffalo, NY 14217
716/874-5510
or 1736 Avenue Road
Toronto, Ontario
Canada M5M 3Y7
416/787-1448

Price: \$29.95

ZXFORTH ads boast the "simplicity of BASIC with the speed of machine code." Although I can see some uses for ZXFORTH, simplicity is not among its virtues.

This operating system/language comes on cassette with a 56-page manual and a 5-page editor manual. Instructions for loading both ZXFORTH and the EDITOR appear on the inside front cover of the FORTH manual. This is about the only tutorial material presented in either book. They are reference manuals, and as such will probably prove too obscure for those with no knowledge of FORTH.

Experienced FORTH users may have no trouble adjusting to ZXFORTH, but most likely will consider the implementation insufficient. The editor uses line-by-line (not screen-oriented) techniques; the badly organized manual is written for the wrong computer in parts (containing many references to ASCII characters not appearing on ZX/TS) and gives examples that do not work.

I always find it difficult to work with any language stored in RAM on a typically ROM-based system, because every time the computer crashes I must reload the entire language in addition to my programs. This shouldn't pose a problem for a nice solid interpreter (how often does BASIC crash?), but ZXFORTH doesn't fit the bill.

To anyone unfamiliar with FORTH, this package bewilders. Suddenly you must understand complicated stack operations, strange terminology and, worst yet, many technical inaccuracies. On the other hand, if you already know FORTH or really want to learn it on your ZX/TS, ZXFORTH will probably work for you. The power is there if you can figure out how to use it.

ZXFORTH works. Because of the interpretation techniques, it is flexible and fast. But if you expect to learn FORTH, look elsewhere for your text. These manuals will not lead you by the hand.

SINCLAIR / REAL TIME CLOCK

ZX80/1
TIMEX 1000

I/O Board with our Real Time Clock/Calendar

Time	Month	Date	Year	Day of Week
23:59:59	12	31	99	7

- Features
- 8 Outputs capable of driving relays
 - 8 TTL inputs
 - Feed through Sinclair Bus connector to allow normal expansion
 - Battery back-up for clock
 - Expandable Ports
 - All software included
 - 90 day warranty

Future Products

- Touch Tone™ encoder/decoder
- Speech and Sound synthesizer

MODEL	DESCRIPTION	PRICE
Built & Bare PCB Tested & Manual		
310	MASTER I/O WITH CLOCK	\$59.95
315		\$24.95
MODEL 310 ACCESSORY BOARDS		
320	A to D and D to A converter	\$47.95
325		\$19.95
330	Wireless control system (BSR™)	\$69.95
340	Solid State AC Relay	\$19.95
SPECIALIZED PARTS FOR MODEL 315		
316	Sinclair edge connector, 46 pin w/key	\$12.95
317	Clock IC and crystal (tested)	\$ 4.95

California residents add 6% Shipping and handling \$ 3.95

Send self addressed stamped envelope for catalog.
VISA/MASTERCARD

JK AUDIO
P.O. Box 3295
Escondido, CA 92025-0580
(619) 741-5132 (24 Hr. Order Line)

VOTEM Lets Your Computer See The World



Analog Interface/Tape Signal Conditioner
(for the Timex/Sinclair Computer)

VOTEM is a complete package consisting of hardware and software that enables your computer to measure, display and record "real world" analog signals. Your computer can monitor any physical phenomenon (pressure, light, temperature, etc.) represented by a DC voltage. A probe is provided for air and liquid temperature measurements.

Your computer becomes a "smart" digital voltmeter and thermometer with storage capability. Use VOTEM in a home energy conservation project to save money and possibly qualify for an energy tax credit.

VOTEM also cleans up the tape signal for reliable LOADS. With VOTEM you can LOAD tapes with a lower volume setting on your tape recorder.

VOTEM requires no mods, does not use the computer's rear expansion connector and works with 1K RAM (or more).

VOTEM is low priced at only \$59.95 (assembled and tested). For an even better bargain the VOTEM kit is only \$39.95. (Requires soldering and approx 2 hrs.) VOTEM comes with a detailed 35-page manual. The manual may be purchased separately for \$5 pp and applied to first purchase of a VOTEM unit. If you are not satisfied with VOTEM return within 15 days for full refund. (Does not apply to kits). Send check or money order plus \$3 for shipping and handling.

Down East Computers
P.O. Box 3096/Greenville, N.C. 27834

for TIMEX/SINCLAIR computers

LOST IN SPACE (uses SLOW)	11.95
UNIVERSAL INVENTORY FILE	16.95
UNIVERSAL MAILING LIST	10.95
UNIVERSAL COIN COLLECTION	10.95
UNIVERSAL STAMP COLLECTION	10.95
UNIVERSAL COMIC BOOK COLL	10.95
UNIVERSAL BASEBALL CARD COLL	10.95
UNIVERSAL RECORD ALBUM COLL	10.95

Each program on cassette + manual. (8K ROM, 16K RAM) Please add \$1.50 for shipping and handling. N.J. residents add 5% tax.

M.C. HOFFMAN CO.
P.O. BOX 117
OAKLAND, N.J. 07436

The Parallel Connection

Product: Memotech Parallel Interface and Seikosha GP100A Printer

From: Memotech Corp.
7550 W. Yale Ave. Suite 200
Denver, CO 80227
303/986-1516

Price: \$408.95 for I/F, cable and printer, including shipping

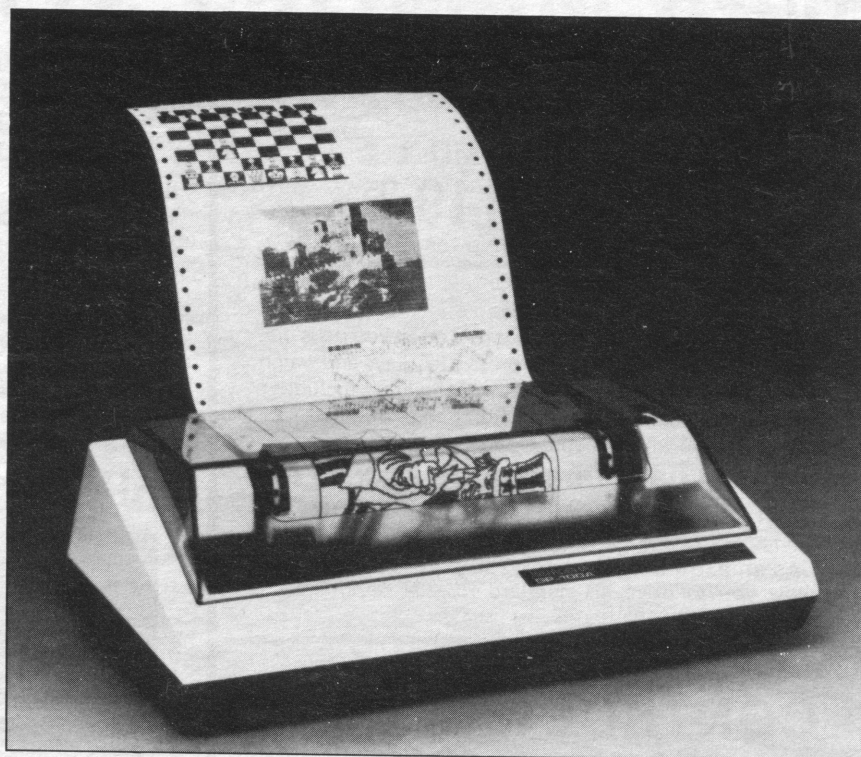
After my large box from Memotech arrived, I examined the components: a Memotech I/F, cable and 80-column printer.

I/F stands for interface, computer jargon for the connection between your ZX81 or Timex Sinclair 1000 and the printer. Memotech's interface is Centronics parallel-equivalent. So any printer that uses a parallel printer port (one of two common standard ports) will work with it. This interface uses decimal addresses 10500 to 11500. With a 64K RAM pack you must only turn off the 8-12K area because the addressing is fully decoded. As with all Memotech products, the I/F sandwiches with other Memotech products (the I/F must plug into the computer before anything else). The product packaging is Memotech's standard black anodized case. The interface is FCC-approved for RF noise.

Then I unwrapped the cable. It's made by 3M and has 34 pins on one end, 36 pins on the other. The 36-pin end goes to the printer.

Finally, the printer emerged. It is the same as Radio Shack's Line Printer 7, which sells for \$399. It is pin-feed only and prints about 30 cps (characters per second), giving 80 columns in a 5x7 dot matrix. The printer runs a little noisy, even with the sound baffle on. You can get all the paper or ribbons you need at Radio Shacks anywhere.

Once I connected everything I was able to print in minutes. The I/F uses all the keyboard's printer commands—no USR calls here. All ASCII control codes are preceded by an inverse period. Or you can use the CHR\$ of the Sinclair code corresponding to that character. Lower



case characters are represented by inverse graphics. For example, inverse B prints as "b".

I have used my printer for a month or so with no problems. The entire package costs \$408.95 and offers a 96-character ASCII character set with lower case, double-wide characters, and dot-addressable graphics. In addition, it can print Memotech's hi-res screen. You get 80-column printouts for LLIST and LPRINT; COPY gives you exact screen image.

For the price, you can't beat it.

Having gained considerable practice with this unit, I will now pass on my experience to you:

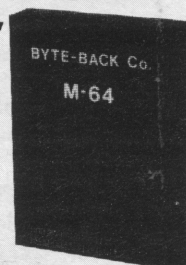
1. Memotech's literature does not go into detail about ASCII control codes. Check at computer stores to locate a good, detailed code guide. Be sure to check all codes against your printer's guide because some codes differ. For example, with the Seikosha printer change the following: inverse period 8—graphics mode; inverse period A—NEWLINE or ENTER; inverse period G—POS.

```
10 REM AAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
15 LET B=16514
20 INPUT A
30 POKE B,A
40 IF A=0 THEN GOTO 100
50 LET B=B+1
60 GOTO 20
100 LPRINT CHR$ 6
```

Program 1

BYTE-BACK modules

64-K MEMORY \$119.⁹⁵_{KIT}



INSTANT INFORMATION WITH BYTE-BACK'S MD-1

MODEM only \$119.⁹⁵_{KIT}

WIRED AND TESTED \$149.95

Use your phone to connect your "LITTLE" computer to the "LARGEST" computer networks in the world. With **BYTE-BACK's MD-1 MODEM** connected all you do is dial a phone number (usually local), press a few keys and watch the data appear on your TV screen. (Software is included) This **MODEM** can be used in either the "originate" or "answer" mode with selectable baud rate.

You can have immediate access to:
UNIVERSITY COMPUTERS, DOW JONES,
COMPUSERVE & SOURCE

As an extra bonus an RS-232 port is provided to allow you to drive all standard RS-232 peripherals & printers. (75 to 9600 Baud)

WIRED and TESTED \$129.95

IN STOCK!

SAME DAY SHIPMENT!

WHY PAY MORE?

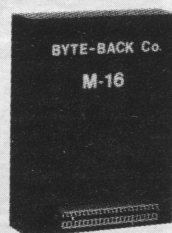
BYTE-BACK'S M-64 extends the memory of your ZX81 or Timex-Sinclair 1000 to a full 64-K. It's user transparent. It plugs directly into the back of the ZX81 and has an expansion port to allow you to still use a printer. No extra power supply is required.

All Memories have extra tight contacts in the connector. This combined with a flat-bottom metal case, provide the shielding and stability needed to **completely eliminate memory "crashes"** common with other memory modules.

EXPAND YOUR 16K SYSTEM

(M-16) \$59.⁹⁵_{KIT}

WIRED and TESTED \$69.95



If you have a Sinclair 16K RAM module and need more memory, expand it to 32K and beyond by using **BYTE-BACK M-16 MEMORY MODULES**.

You can't connect two Sinclair 16K RAM modules together, but you can connect one Sinclair 16K and one or more **BYTE-BACK 16K** modules to get all the memory you need.

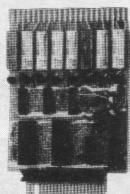
THOUSANDS IN USE WITH PROVEN RELIABILITY

IN STOCK — SAME DAY SHIPMENT

BYTE-BACK'S BB-1 CONTROL MODULE \$59.⁰⁰_{KIT} *In Stock!*

WIRED and TESTED \$69.

- **8 Independent Relays**
(with LED status indicators)
- **8 Independent TTL Inputs**
(with Schmitt trigger buffers)



• By using a single POKE command you can change and latch the status of each of the 8 relays.

• Your computer can read the status of all 8 inputs by the use of single Peek command.

• A comprehensive manual is included that has complete application details.

Don't settle for a poor quality printer . . .

RS-232 Module \$59.⁹⁵_{KIT}

WIRED AND TESTED \$69.95

Allows you to connect to all RS-232 printers & terminals.

ALL MODULES CARRY 90-DAY WARRANTY

Schematic & Manual included with all bd.s and units.

TRY BYTE-BACK MODULES FOR 10 DAYS WITH NO OBLIGATION

Remember with: BYTE-BACK modules you are NOT limited to using only one module at a time!

☐ M-64 Kit	\$119.95	☐ M-16 Kit	\$59.95
☐ M-64 Wired and Tested	\$129.95	☐ M-16 Wired and Tested	\$69.95
☐ M-64 Blank PC Board	\$19.95	☐ M-16 Blank PC Board	\$19.95
☐ BB-1	\$59	☐ Modem Kit	\$119.95
☐ BB-1 Wired	\$69	☐ Modem Wired and Tested	\$149.95
☐ BB-1 Blank PC Board	\$19.95	☐ RS-232 Module Kit	\$59.95
		☐ RS-232 Module Wired and Tested	\$69.95

Shipping and Handling \$4.95

ORDER PHONE (803) 532-5812

Bill My ☐ Visa

☐ MasterCard

Exp. Date _____ Card No. _____

Name _____

Address _____

City/State/Zip _____

Mail To: **BYTE-BACK CO. • RT. 3 BOX 147 • BRODIE RD. • LEESVILLE, S.C. 29070**

DEALER DISCOUNT AVAILABLE



CHECKS



BYTE-BACK CO.
RT. 3 BOX 147 • BRODIE RD.
LEESVILLE, S.C. 29070
Ph. (803) 532-5812
BYTE-BACK CO. SUPPLIES COMPUTER EQUIPMENT
& MODULES TO: NASA, U.S. GOVERNMENT
AGENCIES, BELL LABS, G.E., MARTIN-MARIETTA,
ALLIED CHEMICAL, & MANY SCHOOLS & UNIVERS-
ITIES.

2. When using LPRINT and TAB you can only TAB 32 positions. To use the printer's full width, substitute LPRINT "inverse period Gxx"; "Data" where xx is 0 to 80. This prints at character position xx.

3. Instructions from Memotech do not directly tell you how to use the printer's dot-addressable graphics feature. But one part shows how to POKE codes into a REM statement and print symbols for these values. That's how.

To demonstrate, key in program 1. Line 10 contains 50 repetitions of the letter A (any character would do). Line 15 starts at the first memory address after the REM in line 10. A in line 20 is the decimal

number you will enter. Line 30 puts each decimal number in a memory location. Line 40 checks for a zero input (0 tells the subroutine to exit and print).

Now RUN the program. Enter each decimal number following (hit NEWLINE or ENTER after each):

- 8 (puts printer in graphics mode)
- 27 (ESCape, when followed by POS specifies dot-address)
- 16 (POSition)
- 1 (High Point)
- 100 (Low Point)
- 2, 255, 186, 160, 178, 191, 178, 160, 178, 191, 178, 160, 186, 255, 2 (graphics data—see manual)
- 0 (tells subroutine to exit and print).

High Point (HP) and Low Point (LP) combined form a 4-byte dot address that can range from 0 to 479 decimal or 01DF hex (the dot-address of the last character). Here is the snag. HP must equal either 00 hex (if the address is less than 256) or 01 hex (if the address is greater than or equal to 256). Yet if you enter 00, the subroutine exits and you get no print. The printer only checks whether bit 7 of the HP is on or off. Instead of 00 hex, enter 16 decimal. For 01 hex enter 01 decimal. LP can be any number between 0 and 255 decimal, or 00 and FF hex.

Jeff Pack
Auburn, WA

Fiz-zy Floppy

Product: Floppy Interface for ZX81 (FIZ)

From: Macronics Systems Ltd.

26 Spiers Close

Knowle, Solihull

West Midlands, UK B93 9ES

Price: \$149 for card only, \$35 for expansion bus. Shipped airmail at no extra charge.

Macronics' is the first disk drive interface that appeared for testing with ZX81 computers. The interface card (5x5½ inches) fits onto a motherboard so you can use a printer and external RAM pack. Any RAM pack that lets you switch off the 8-12K region of RAM will work with this system.

You can supply your own disk drive or get one from Macronics. There is a standard 5¼ inch floppy drive. (According to Ken MacDonald of Macronics, their 3-inch drives should be available around March, 1983. In the US, a Shugart SA400 or equivalent can work with this interface—Ed.) A 21-inch cable couples the drive to the interface card. This cable is more than ample to move the disk unit into an easy working position.

You switch on the disk drive before applying power to the computer. Here I ran into my first problem—there is no way to tell that everything is alright. Usually, a disk system's software first tests the drive so you hear the motor when you switch on the power. Not so on the Macronics drive. You might expect to see the LED on the front light when the power comes on. Not so. This LED only comes on when the computer uses the drive. In fact, this software only responds to USR calls; otherwise it is inactive.

To use this system, you need single-sided, single-density, hard sector disks. These disks are divided into 10 sectors. Because each disk contains 35 or 40 tracks, you get 350 or 400 segments of data, but each file must use a whole number of tracks. With 35-track disks, you get 42.5K bytes maximum storage; 48.75K with 40-track disks. Each data segment is 128 bytes long. The whole first track is used for the directory. This leaves you with no more than 48K on each disk and no more than 39 files per disk. Of course, you can swap disks during a program for greater storage capacity.

Disk systems' speed usually attracts people, but this one is SLOW compared to most. Software entirely controls the disk—there is no controller board.

This system takes 24 seconds to load a 16K program from disk into memory. (MacDonald says it takes 18 seconds—Ed.) Although this speed is 21 times the speed of cassette loading, it is six times slower than Sinclair's forecast ZX Spectrum Microdrive and 60 times slower than normal disk drives for other computers.

Despite the slow speed, Macronic's disk system is ideal for ZX/TS users. It is simple to operate and understand.

Technical Details

Files always use a whole number of tracks and your program must discern what data is in each sector of the file. The disk system provides 10 commands that you can use in programs via one USR call to Macronic's 4K ROM on the interface board. This command, LET E=USR 9999, should be the first line of any program. All other commands are then

available as `USR` calls to the ROM using letters or words instead of numbers. For instance, `USR DIR` calls the directory (2-second access time), which prints disk contents on the screen by file name only. `DIR` has no effect on your program except to clear the screen. You can `COPY` the directory listing.

Each file name consists of six letters, no spaces, followed by a full stop (period). Then comes a three-letter description of the file type (`DAT` for data, `BAS` for BASIC, for example). You can invent your own codes if you like.

To create data files, you must first give variable `F$` the name of the file in the format just described. Then assign the number of sectors required (in multiples of 10) to variable `R`. The command `LET E=USR CREATE` will then make a new file. If a file of the same name exists, the system will not overwrite it, but return the variable `E` with an error code. You can only copy a file by giving it another name and only destroy it by a `KILL` command.

`READ` and `WRITE` use variable `R$`, which has been dimensioned to be 1x128 bytes long to read and write sectors into and out of memory. You use `R` for the number of sectors required and `F$` for the file name. `R` may not be a `FOR-NEXT` loop variable.

`STAT` gives the disk status. If `F$=""` (no file name), `STAT` gives the number of free sectors on the disk. If `F$` contains a file name, `STAT` returns the number of sectors that file uses.

`DSAVE` and `DLOAD` do the same as on the cassette, so you can `LOAD` files into the computer from tape and `DSAVE` them onto disk. You can also make programs auto-run.

`NEWD` initializes a new disk, filling every sector with check characters and writing out the directory track.

The supplied interface card has 2K RAM for use as work space for the disk at 14-16K. It also contains a 4K ROM using addresses 8-12K. (According to Ken MacDonald, the `FIZ` uses 240 bytes of ZX/TS RAM in the variables region—Ed.)

Conclusions

This disk system in no way rivals Sinclair's Spectrum Microdrive (when available). Sinclair's Microdrive will not work with ZX81s or TS1000s and will beat Macronics' drive both in capacity (100K per disk, eight drives) and price (about £50, or \$82 US).

However, the ZX81 (and TS1000) is well used all over the world and the Spectrum has been only available in the UK. ZX/TS users who do not want to change their machine and software will find this a very useful, but expensive way to speed up data processing.

There are a few things I'd like to see Macronics change:

1. An at-a-glance list of commands would be better than leaving the commands spread throughout the manual.
2. A `POWER ON` indicator would be useful on the disk drive.
3. Arrangements should be made to have 48K RAM packs available for use with the system.
4. Use `DIM R$(1,128)` instead of `DIM R$(128)`. This would allow users to `INPUT` directly to the variable used by the disk and save memory.

Stephen Adams
London, England

Keyboard Quickness

Product: Auto Shift Keyboard Kit for ZX/TS

From: Research Applications
Products

4561 Paloma Lane
Yorba Linda, CA 92686

Price: Kit \$80
Board and Key Legends \$30

I picked up my keyboard kit at Research Applications, so I did not have to wait for mail order delivery. Although you can buy the PC (printed circuit) board and key legends for \$30, I opted for the complete kit. Compared to rounding up the remaining components, the savings didn't seem worth the hassle.

One double-sided PC board, 73 keys and key caps, 7 CMOS ICs and sockets, 27 resistors, 27 diodes and mylar key legends make up the kit. It does not come with an enclosure, but this was no problem—it fits easily into a surplus enclosure I bought locally for \$6.

Construction time was about four hours—there are 134 components and 73 separate key legends to mount.

Keyboard connection to the computer is via ribbon cable which you

solder to the board at one end. You then strip and tin the connectors on the other end before inserting them into existing sockets on the computer. (The connectors are on 0.1 inch centers and slip directly into the sockets.)

For ease of use, I replaced the kit's ribbon connector with a five-foot length, which resulted in the autoshift `LPRINT` not working. Fortunately, the instructions addressed this problem. I easily solved it by adding a 6.8K resistor in parallel with the 10K pull-up resistor on the computer `KBD0` line.

My completed keyboard is professional in appearance and extremely easy to use. As advertised, 46 key functions are automatically shifted. The keyboard layout is logically arranged with a special graphics section. The editing/game keyrow (located at the bottom of the keyboard) is a nice touch. This feature gives you quick and easy program editing with single keystrokes and contains the cursor arrows used in many games.

Don Parks
Riverside, CA

Pocket Information

Title: *The ZX81 Pocket Book*
 Author: Trevor Toms
 From: Reston Publishing Co., Inc.
 599 Adamsdale Road
 N. Attleboro, MA 02760
 Price: \$10.95

For a 128-page paperback, \$11 is a lot of money to pay. But Trevor Toms has done a good job here. There are a couple dozen programs and subroutines, mostly games but also a few handy utilities: big characters, a hex monitor, and a couple of very nice routines for formatted numeric output are included. All the programs I tried were good to excellent.

There is also a good deal of text aimed at the moderately experienced user who wants to develop techniques for running faster programs and using less memory, and methods and algorithms for doing things that aren't completely straightforward in Sinclair BASIC. Anyone past the beginner level will find some useful ideas in this book. For instance, which is better: $A = B * 0.15$ or $A = B * 100 / 15$? Toms points out that while the first is faster, it turns out to be less accurate; $INT(100 * 0.15)$ returns a value of 14!

Minor errors mar the book. Toms originally wrote the book for an English audience, and this American edition lacks some necessary corrections. For example, the statement that FRAMES is updated at intervals of 1/50 of a second was not changed, though in the American ZX/TS computers it's 1/60 second. In addition, in the section on reducing execution times, some benchmark programs are listed and it says that there is a table of timings at the end of the book. There is not. Remarkably, I found no bug or typo worth mentioning in any of the programs—even the long and complicated "Adventure Master" given in the last chapter.

This marvelous program is, in effect, an interpreter for a language to create adventure-type games. Toms

gives two sample scenarios, a tiny test scenario and a more complex one, "City of Alzan." The latter is fairly simple but fun, and it doesn't begin to strain even a mere 16K RAM's capacity. If you'd like to make your own clever and complicated adventure game, this program

should give you all the tools you'll need. This chance to create your own adventure game may justify the price of the book.

Richard S. Holmes,
 Batavia, IL

Know Your CPU

Title: *Z80 User's Manual*
 Author: Joseph J. Carr
 From: Reston Publishing Co.
 A Prentice-Hall Co.
 Reston, VA
 Price: \$10.95 (326 pp., softcover)

In a nutshell, the *Z80 User's Manual* is an excellent reference but not an instruction manual for beginners.

While most Z80 hardware technical manuals tell you what you need to know about any given facet of the computer's operation in five different places, Carr's *Z80 User's Manual* tells you everything you need to know in one place. For instance, in the description of the memory refresh register R, this manual tells not only what the R register does, but that the I register is placed on the high order bits of the address bus, and R register on the low order bits, and that the R register does not increment through the eight bit—all important information when trying to understand ZX/TS character generation and keyboard scanning. This sort of information may be split up in other manuals, or may not be presented at all. I wish this book had been around when I learned Z80 architecture and programming!

To understand the hardware section, you should have some familiarity with digital logic, since it is used extensively.

The hardware section includes interfaces for RS232 and current loop devices, I/O port interfacing, mem-

ory mapping, a good description of interrupts, explanation of ASCII and EBCDIC codes (neither of which the ZX81 uses), and short descriptions of the Z8 and Z8000 microprocessors, which will give you a more worldly view of microprocessors.

The software section of the manual describes how addressing modes work, interrupts from a software standpoint, and, to some degree, coding techniques. In the software portion of the book, Carr assumes you understand logical operators (AND, OR, etc.); he gives no explanations while using the terms.

The last chapter is a comprehensive description of the Z80 instruction set, similar to those published in any book on Z80 microprocessors, although a little better done than most.

This book is not for beginners with no experience or concept of how machine code works. It may also be over the heads of those just beginning in digital circuitry. It is a very good reference work for those who are eager to learn.

Carr presents concepts in a clear, concise form, including everything you need to know on a subject in one place. This manual does not tell you amazing things people didn't know before; it retells the old story in an understandable form. A warning, however: this book does not deal specifically with the ZX81 at all, and most of the circuits given will require some modification. In general, the circuits are shown as concepts, and therefore will not work as is.

SYNTAX

SERVING TIMEX-SINCLAIR
PERSONAL COMPUTERS

A PUBLICATION OF THE HARVARD GROUP

ISSN 0273-2696

SYNTAX is a monthly newsletter exclusively for ZX80/81/TS1000 owners. We bring you news, reviews and applications for your computer, plus technical notes for circuit-builders. SYNTAX also provides a forum for thousands of users to share advice and problems about programs and vendors. We bring you timely updates about new hardware, software and books. And we cover all the Sinclair-technology computers, including the new TS1000.

At SYNTAX we emphasize practicality. You can apply our suggestions even if you aren't sure at first why they work, because we give you complete instructions. Text is clear and easy to understand. SYNTAX readers already know about:

- An automatic phone-dialer they can put together in a few hours
- Syntactic Sums™ to check input for errors
- Programs to explore computer memory
- How to build external additional RAM
- How to add an 8212 I/O chip to control external devices from their computers

And SYNTAX readers like what they get every month. Subscribers know they can depend on us.

After receiving only three issues of SYNTAX, I find that I anxiously await the next... keep up the good work!

Martin Irons
Goshen, NY

Congratulations on the brass-tacks, down-to-earth approach of your newsletter. I'll be looking forward to future issues.

Otis Imboden
Washington, DC

Many readers get their first issue and immediately order the back issues—more proof that they like what they see.

You can see what's special about our publication. We work hard to bring you a quality newsletter. We strive to print useful programs of above-average accuracy. As any computer magazine editor can tell you, program listing accuracy is tough to achieve, but we boost our average with every issue. We test each program to make sure it works, it fits in the designated RAM, and it runs when you follow the directions. We print program listings in screen-image format to make it easier for you (it's sure not easier for us!) to enter program accurately. We invented Syntactic Sum™ as an additional aid for you in getting error-free programs. With your subscription you also get access to thousands of other readers, and our staff experts are available by phone to answer your questions or help you solve problems with your machine.

SYNTAX readers get every month:

- Latest news of Z80 hardware and software
- Programs to organize information, calculate, entertain, or instruct
- Do-it-yourself additions
- Clear explanations for beginners

To share the benefits of SYNTAX, just indicate your choices on the order coupon and return it with your choice of payment in U.S. funds. (Please note that additional postage is required for delivery outside North America.)

We are so sure you'll find SYNTAX useful that we promise to refund your entire subscription fee if you aren't satisfied. An unconditional guarantee—you can't lose.

Join the others who stretch the ZX/TS to the utmost. Act now—as soon as we receive your coupon with payment, your first issue will be on its way. For faster service, phone your credit card order to 617/456-3661. Don't miss SYNTAX!

Fill out the coupon below and mail it to: SYNTAX/SQ, RD2 Box 457, Harvard, MA 01451

SQ383

☐ My check is enclosed.
Make checks payable to:
SYNTAX ZX80, Inc.

☐ Please charge my ☐ VISA
☐ Diner's Club ☐ American
Express ☐ Mastercard
☐ Carte Blanche account

Account number _____

Exp. date _____ Bank number (MC only) _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Day phone () _____ Evening phone () _____

☐ This is a renewal. My subscription number is: _____

☐ This is a new subscription.

YES! Please send me:

- | | |
|---|--------|
| ☐ The Combination (12 issues of SYNTAX and 4 issues of SQ) | \$39 |
| ☐ The Catch-up (SYNTAX Jan. 82-Dec. 83, 4 issues of SQ, plus 1 binder) | \$77 |
| ☐ The Works (SYNTAX Nov. 80-Nov. 83, 4 issues of SQ, plus 2 binders, a 50% savings) | \$97 |
| ☐ 12 issues of SYNTAX | \$29 |
| ☐ 4 issues of SQ, The Syntax Quarterly | \$15 |
| ☐ 1 Premier Issue of SQ | \$4.95 |
| ☐ 1 Binder | \$9 |
| ☐ 1 issue of SYNTAX | \$4 |

These offers expire 31 May 83—SUBSCRIBE NOW.

Increase Your Computer's Utility With Our Help

You own a powerful computer, capable of sorting, analyzing, calculating, displaying and manipulating data as well as measuring and controlling. Despite its small size, your computer will help you learn, analyze your business, keep your records or your schedule, dial your phone, send messages, or talk to other computers. To control the power available to you, you'll need hardware and software to use with your machine. Although you can buy many of these products—and we'll tell you about them—some you must create or modify for yourself. SQ will provide you with the complete information to let you use your programming or tinkering time efficiently.

Our experts also bring you SYNTAX, the newsletter, for up-to-date, concise information. You need news and new product announcements quickly. And SYNTAX packs a lot of information into its brief, time-saving format.

But we recognized your need for material too long to fit in SYNTAX. Syntax Quarterly fulfills that special need. In-depth, extensive, detailed information—that's what SQ's all about. Long programs, big construction articles, and detailed directions help you use your computer to its fullest while learning.

SQ gives you tested, accurate information that you can depend on. We test, we build, we check everything that goes into SQ. Our staff experts work for you; it's as though your full-time staff prepared a report for you every quarter.

At SQ, we specialize in your machine—Timex-Sinclair technology. Our programs and projects work, and, because we test each one before we publish it, they work on your machine. All the information in SQ can work for you.

But, to capture these benefits—free software, free plans for hardware, free instruction—you must subscribe. Use the coupon to tell us how to help you best. Or call; we understand people in a hurry.

SQ

```

100 LET A$=" "
200 LET B$=" "
250 LET C$=" "
300 PRINT A$,A$
400 PRINT B$,B$
500 PRINT B$( TO 7),B$
600 PRINT B$( TO 7),B$
700 PRINT A$,B$
800 PRINT " ";B$(2 TO ),B$
900 PRINT " ";B$(2 TO ),B$
1000 PRINT B$,B$
1100 PRINT A$,A$
1200 PRINT AT 7,22;C$
1300 PRINT AT 9,24;C$
1400 FOR I=0 TO 248
1450 IF I>66 AND I<128 THEN GOTO
1600
1500 PRINT CHR$ I;
1600 NEXT I
1700 PRINT AT 10,0;" WE SPEAK Y
OUR LANGUAGE
SYNTACTIC SUM: 22174, 8K ROM
  
```

SQ

WE SPEAK YOUR LANGUAGE 0123
456789ABCDEFGHIJKLMNPOQRSTUVWXYZ
RNDINKEY\$PI
0123456789ABCDEFGHIJKLMN
PORETUNNYE"AT TAB ?CODE VAL LEN
SIN COS TAN ASN ACS ATN LN EXP
INT SQR SGN ABS PEEK USR STR\$ CH
R\$ NOT ** OR AND <=>=<> THEN TO
STEP LPRINT LLIST STOP SLOW FAST
NEW SCROLL CONT DIM REM FOR GOT
O GOSUB INPUT LOAD LIST LET PAUS
E NEXT POKE PRINT PLOT RUN SAVE

617/456-3661

617/456-3661

Inside Story

Title: *Secret Guide to Computers*
 Author: Russ Walter
 From: Secret Guide to Computers Company
 92 Saint Botolph St.,
 Boston, MA 02116
 617/266-8128
 Price: \$3.70/each or 8 vol. set for \$29.60

The tenth edition of Russ Walter's *Secret Guide to Computers* is probably the funniest, most interesting, most entertaining computer learning guide ever written; however, it contains little on the ZX/TS computers. Learn the ZX's BASIC first, then buy the guides.

Here's a rundown of all eight volumes of the *Secret Guide to Computers*. Volume 1 starts the reader off with a lesson on the keyboard then proceeds to fixing mistakes, math, variables, input, program storage, logic, and all the other necessities so you can get started with BASIC on most micros, some minis, and some maxis.

Deeper into BASIC

Volume 2 digs deeper into BASIC. It includes information on ON...GOTO and ON...GOSUB statements (something the ZXs don't have), error handling, error trapping, simpleton graphics, advanced fundamentals, programming style, math functions, sorting, and text files. This volume is the most helpful for ZX users since neither the ZX80 nor the ZX81 manual mention anything about graphics, fundamentals, or style and sorting.

Volume 3 covers hardware, buying a computer, computer companies, peripherals, explanation of the Central Processing Unit (CPU), and a rundown of the major microcomputers.

The Computer Field and Basic Applications

Volume 4 digs deep into the computer field. It rehashes the major microcomputer manufacturers, and goes over the major points of buying a big computer. This volume also covers the different number systems,

getting a job in the computer field, what to read, and computer courses offered by the author.

Volume 5 covers some basic applications. For example: games, extensive graphics, and the all-important VISICALC and its mates SUPER-CALC, SPECTACULATOR, and word processing in most micros.

Volume 6 isn't available but I contacted the author who said it will cover artificial intelligence and expand on Volume 5.

Volume 7 covers FORTRAN, PASCAL, COBOL and Volume 8 goes into an extensive overview of all the computer languages but doesn't teach you how to use them. Volume 8 also goes into the history of the CPU and how to program in assembly language for most of the CPUs. Yet, I suggest buying Dr. Ian Logan's book written especially for the ZXs because Volume 8 doesn't tell you how to go about assembling. It just explains various mnemonics.

Overall, Russ Walter's eight-volume guide is both fun and very informative. Most computer books are too technical, but the *Secret Guide* is written in plain English.

Peter Nichols, Hanover, MA

Ad Index

Brainchild Computer Works	52
Byte-Back Co.	58
Compuwiz Software	36
Dallas Development	21
Down East Computers	56
E-Z Key	37
Ezra Group II	36
Frog Software	64
General Systems Consulting	23
Gladstone Electronics	13
M.C. Hoffman	56
Innovation Company	C3
Intercomputer, Inc.	17
International Publishing & Software	1
JK Audio	56
Kopak Creations, Inc.	C4
Melbourne House Software	33
Memotech Corporation	7
Mindware, Inc.	C2
Sinware	6
Softsync, Inc.	18
Syntax/SQ	3, 16, 62, 63
Tom Woods	64
Z-Ware	64

Z-Ware Sale

ZX81/TS-1000 Software on Cassette
 \$8.95 each ppd.
 16K Histogram—plots bar chart
 16K Financial Manager—for home or business
 16K Machine Intelligence Demo—life forms that actually learn.
 Many more available—send self-addressed, stamped envelope.

Z-Ware
Box 111
Albany, Kentucky 42602
1-606-387-8391

LEARN TO PROGRAM

Text and File Organizers with ZX DATA FINDER

A high capacity information storage and retrieval tool for 16 K Timex and Sinclair Computers.

Advanced file input and editing routines are thoroughly analyzed.

Comprehensive search and display methods are fully explained.

AN ADVANCED COURSE IN DATA HANDLING

Free specifications are available, or send \$9.95 for program listing and text to:

THOMAS B. WOODS
P.O. Box 64, Jefferson, N.H. 03583

ARTIFICIAL INTELLIGENCE FOR THE ZX81? YES!

SYNCZX is an artificial intelligence program with natural language capabilities for the ZX81 with 16K RAM, available from Frog Software.

SYNCZX will talk to you in ENGLISH/NO MENUS. You can use SYNCZX to balance your checkbook or you can reprogram SYNCZX to do anything you would like. However you do not need programming skills to use SYNCZX as is. Even a child can use SYNCZX because it is easy to read and understand, talking to you in simple English. SYNCZX even remembers the people who use it! 16K cassette with manual only \$6.95 plus \$1.50 postage & handling.

Frog Software

Box 95
Glenmont, New York 12077
(518) 465-6552